The design and evaluation of secure systems: An approach using patterns

Eduardo B. Fernandez, Florida Atlantic University, Boca Raton, FL, USA

Chapter 1. Introduction

Information is the currency of democracy. --Thomas Jefferson, third US president, architect and author (1743-1826)

We will bankrupt ourselves in the vain search for absolute security.
Dwight D. Eisenhower, U.S. general and 34th U. S. president (1890-1969)

1.1 The value of information

In our everyday lives we depend heavily on computers; most of the devices that make our life safe and convenient, e.g. cell phones, vehicle controls, and building controls, use some type of software. We also rely on institutions, public or private; we are born in hospitals, then go to schools, join clubs, get jobs in the government or private businesses, get married at some church or public office, travel using some agency, etc. All these institutions use computerized information, a good part of which is about us. Clearly, the data used in devices has a significant value; cell phones contain personal information, for example. In general, the data of an institution has great value; it may represent customers, orders, bills, business plans, course grades, etc. For example, data corruption in a hospital may result in patients getting the wrong medication, leakage of military information could endanger an army in war, and erroneous aircraft maintenance information could compromise passenger safety. We don't want the data of these institutions to be seen by those who could misuse it; enterprises want their information to be hidden from competitors, and patients don't want their medical records to be seen by unauthorized people. Data and other resources are *assets*, items that have value for us*;* security is the protection of assets, including enterprise and individual information.

We need this protection because there are people who intentionally try to access or modify information either for their own gain, for political purposes, or for the sake of disruption. Some of their actions include viruses and similar attacks that corrupt information; some actions intend to access or modify information. In addition to the direct monetary cost there may be losses of productivity, and even endangering of lives. According to the Ponemon Institute [Pon17], the average total cost of data breach for the 419 companies participating in this research decreased from $4.00 to $3.62 million. The average cost for each lost or stolen record containing sensitive and confidential information decreased from $158 in 2016 to $141 in this year's study. However, despite the decline in the overall cost, companies are having larger breaches, the average size of their breaches increased 1.8 percent.

Security is the protection against:

- Unauthorized data disclosure (*confidentiality or secrecy*).
- Unauthorized data modification (*integrity*). Unauthorized modification of data may result in inconsistencies or erroneous data. Data destruction may bring all kinds of losses.
- Denial of service—Users or other systems may prevent the legitimate users from using their system. Denial of service is an attack on the *availability* of the system.
- Lack of accountability—Users should be responsible for their actions and should not be able to deny what they have done (*non-repudiation*).

This list defines the *objectives* of a secure system: how to avoid or mitigate the effect of these misuses of information. These aspects, sometimes called security goals or properties, try to anticipate the possible risks of the system, that is, the types of attacks against which we have to be prepared. These generic attacks take specific forms, which we describe in detail in Section 1.7. Different applications may emphasize one or more of these aspects over others. For example, the military worry more about confidentiality, a financial institution may emphasize accountability and integrity, infrastructure systems usually emphasize availability.

The generic attacks mentioned above are realized in two basic ways: by *direct attacks* from a person trying to exploit a vulnerability or flaw in the system, or using *malware*, software that contains code that exploits one or more of these flaws. We need to find *defenses* or *countermeasures* against these attacks. Security defenses may operate in one or more of three modes and any effective strategy must combine these three aspects. First we should try to *prevent* or *mitigate* an attack using some appropriate mechanism. Prevention means completely stopping the attack while mitigation implies partial defense or reducing its effects. If we cannot stop or mitigate the attack, at least we should be able to know that an attack is happening, i.e., we should *detect* the attack. Detection is also useful to stop an attack because it can alert other mechanisms to take action. After the attack has happened we need to have ways to *recover* from its effects and analyze it so we can improve the system. During the attack, *logging* of data access can help reconstruct the data and analyze the attack as well as determine possible perpetrators (*audit*).

The definition of security above describes security as defense against some types of attacks. The generic types of defenses (also known as *countermeasures*) that we can use include:

- *Identification and Authentication (I&A)*—Identification is a user or system action where the user provides an identity. Authentication implies some proof that a user or system is the one he/it claims to be. The result of authentication may be a set of *credentials*, which later can be used to prove identity and may describe some attributes of the authenticated entity (See Chapter 4).
- *Authorization and Access control (A & A)*—Authorization defines permitted access to resources depending on the accessor (user, executing process), the resource being accessed, and the intended use of the resource. Access control requires some mechanism to enforce authorization (See Chapter 2).
- *Logging and Auditing*—These functions imply keeping a record (log) of actions that may be relevant for security and analyzing it later. They can be used to collect

evidence for prosecution (*forensics*) and to improve the system by analyzing why the attack succeeded.

- *Hiding of information*—It is usually performed by the use of cryptography but steganography is another option (See Chapter 3). The idea is to hide the information in order to protect it.
- *Intrusion detection*—Intrusion Detection Systems (IDS) alert the system when an intruder is trying to attack the system (see Chapter 7).

These defenses are sometimes described as *security services* because they are embodied in some subsystem that provides the corresponding functions.

Security attacks are intentional attacks from people having some goal. This makes the work of the defender harder because she must anticipate all possible attacks. The attacks come from external attackers and from insiders. An insider can be a strong risk to an institution because of her knowledge of the system and her access to internal information.

Assurance is a measure of the trust we put on a system as being able to provide the required degree of security. The system must be able to enforce any restriction we define in it, possibly using a combination of hardware and software. *Risk analysis* is the study of possible attacks and their impact in terms of money and time. We need it to perform some risk analysis in order to decide on the investment on systems and personnel to provide security. In particular, we need to enumerate the threats to the system to know how much effort and money we should invest to make the system secure. We will study here how to identify attacks but not their economic impacts. *Governance* refers to the responsibilities of the institution directors to define policies and regulations to handle their information. Typically directors focus on increasing the value of their assets but don't pay much attention to their security. According to statistics, more than 50% of the businesses in the world spend 5% or less in their security budget.

There are three aspects to security:

- Physical security. Control of access to computer installations, buildings, and protection of physical assets.
- Administrative or managerial security. Related to personnel screening, operating procedures, training procedures, and reaction to losses. Many breaches of security occur because of the lack of knowledge of users and administrators.
- Information security. Concerned with hardware and software aspects of data protection. It can also be used to control access to physical units. An important aspect is the definition of security policies that will guide the design of the security system, called *governance*, and which intersects with administrative security.

The purpose of this book is to study technical aspects of information security, that is, we will consider only attacks using the software or hardware of the system. Managerial and physical security aspects are as important, and needed for a secure system but are outside the scope of this book. Part of physical security is converging with information security and some of the defenses we study also apply to it. We emphasize a conceptual engineering approach, not an enumeration of practical details or theoretical analysis of security. We want to understand how the system architecture and its components work together to provide security. You might be intending to become a security expert or you just may want to use properly security systems to protect your information or you may need to make decisions where security is an issue (see exercises 1.3 and 1.4 for example). In all these cases a conceptual perspective is useful. There is an increasing need for security experts, take a look at [How08] for an industrial perspective on their needs and what does it take to be an expert.

1.2 Related topics and misconceptions

Several other aspects have an impact on the security of a system. The security of a system is determined mostly by its software. The hardware must provide a basic support (see Chapter 4), but almost all the attacks go to the software. This means that software engineering and software architecture are an important background for somebody trying to build secure systems. The *architecture* of a system defines the system in terms of components (units) and of interactions between these units. Architecture includes: system topology and organization, decomposition into components, assignment of functionality to components, component interactions, system properties (nonfunctional requirements, e.g. performance, security), correspondence between requirements and units. Both the global architecture and the implementation of each of its units have an effect on the security of a system. A global view is also important because security is not the only requirement for an application, we also need performance, reliability, usability, and other features. All these aspects must be balnced and tradeoffs made according to the objectives of the system.

Among the objectives of security we mentioned availability. In general, *availability* is a measure of how long a system is running (maybe not continuously) and usable. Availability may decrease due to accidental errors. Denial of service is an availability attack, and it is usually considered as part of security because of its intentional nature. *Fault tolerance* refers to approaches to make the system able to withstand failures. Failures could expose data and some techniques used for fault tolerance are also useful for security. The combination of availability, security, and fault tolerance is called *dependability*. When we focus into the ability of a system to keep functioning in despite of serious disruption, we enter into the realm of *survivability* [Mel98].Another relevant topic is *privacy*, the ability of individuals to control information about them. Privacy has ethical, legal, social and technical facets, although frequently this word is misused to denote confidentiality; we discuss it in Section 1.4. There are also several other ethical or legal problems related to security, such as how much authorities can violate privacy in the name of security or work efficiency [Tav04]. We also mentioned above integrity. In addition to security, data integrity can be compromised by accidental errors, e.g., system failure, user error, uncoordinated concurrent access. These types of integrity problems are handled by other mechanisms, typically part of database systems, and are not discussed in this book.

Another related aspect, also typically handled by database systems is *recovery*, how to keep back up copies to reconstruct destroyed or modified information in a convenient way. However, recovery does not help against confidentiality attacks. Security may be used to protect intellectual property, the so called *Digital Rights Management* problem, discussed later. Since most of the systems considered here are software systems, the methods of software engineering are very important to produce secure software. A new and related area is *digital forensics,* the study of how to collect and analyze evidence of a computer attack. Important for security is the concept of *trust,* the dependency or confidence we put on people or systems. In any security mechanism we need to trust in some part of the system or in some user.

Lack of knowledge about security has brought a variety of misconceptions. The biggest is thinking that our place, company, system does not have a security problem, that nobody would be interested in us. Another is the idea that security is a network problem. In fact, only about half of the attacks to a system come from outside. Also, network attacks are usually attacks to the operating system and its utilities, not to the messages moving through the network. Related to this is the misconception that security involves mostly cryptography. While cryptography is important, it doesn't solve most of the current security problems. Others appear to believe that all you need is a good firewall. Some people, including some vendors, think that security comes from *patching,* you discover an error and you patch it. Needless to say, patching corrects an error but may bring a new error. We cannot find all problems during the design of a complex system, so patching will always be needed, but we should try to build good quality systems from the beginning.

1.3 Privacy

We defined privacy as the right of individuals to decide about the use of information about themselves. This right is recognized by all civilized societies and is considered a fundamental human right. The first national privacy protection law was the Swedish Data act of 1973. This was followed by the US Privacy Act of 1974. The intent of this act was to protect individuals against invasion of privacy by the Federal Government. This law is complemented by the Computer Security Act of 1987, which defines requirements for federal agencies about the security of their information. A summary of laws as well as a general discussion of privacy principles is given in [Fer81] and [Sum97].

While there are laws to protect privacy against government intrusion, in the U.S. there is little to protect people against business collecting private information, except their voluntary decisions. The widespread use of "cookies" contributes to this situation [Ber01]. A *cookie* is a small file that keeps track of a user's visit to a web site (more of this in Chapter 8).

As a result of the September 11 attacks privacy has been affected negatively in the US. Already in 2000, the Carnivore project [Sob01] gave the FBI free rein to snoop on Internet communications. Later, US airlines were required to turn over to the FBI millions of passenger records, including names, addresses, travel destinations, and credit card numbers [Sch04]. In another incident, the Justice Department requested abortion records from many Planned Parenthood clinics around the country [Pea04]. The Patriot Act was another attempt to limit privacy in the name of fighting terrorism. The Internet and its social networks have also had a negative effect on privacy.

Obviously, privacy laws mean nothing without having ways to enforce their restrictions. Their approaches of data security are then fundamental to any legislation. It should be noted that attempts to make information more readily available for improved use may have a negative effect on privacy. For example, in the US the Health Insurance Portability and Accountability Act (HIPAA) regulations require the computerization and exchange of medical records over the Internet to improve service to patients. However, this easier availability makes this data also easier to be accessed by unauthorized parties. Unless there is a correlated investment on security, privacy will be negatively affected.

1.4 Problems with current systems

Recent surveys have shown some scary facts: most computer systems connected to the Internet are infected with some type of malware which are spreading to other systems; it takes an average of less than 6 days from the moment a vulnerability is announced until some attacker actually uses it; there are more than 30,000 computers (bots or zombies) that are under control of hackers; more than 16 per cent of attacks are targeted attacks as opposed to random unstructured attacks. What is worse these numbers keep increasing. Notice also that malware is just one type of attacks, there are also specific attacks to access databases with the purpose of reading or modifying information.

There are a variety of reasons to account for the frequent successful attacks to current systems:

- *Lack of user awareness and training*. Many users are easy to dupe and open suspicious attachments, answer offers of supposed prizes, etc.
- *Lack of modularity*. Some vendors build systems that cannot be decomposed. These systems lack well-defined interfaces. This prevents checks across the interfaces and a vulnerability in one place propagates its effects to other places.
- *Lack of understanding of secure system design principles*. As discussed in Chapter 2, some of these were formulated as early as 1975. In many places, the approach used to develop secure code is to look for specific vulnerabilities in the code that, while effective, is a very difficult task because of the amount of code.
- *Lack of good design in software development*. The code of many current systems has been built using procedural methods and using languages such as C or COBOL. Modern approaches such as object-oriented programming are still not widely used. Without proper software design, it is very difficult to build dependable complex systems. For example, the early network protocols that are now part of the Internet were not designed for security.
- *Relying on correction by patching*. When a vulnerability is detected, the vendor issues a patch to stop it. This has the side effects of disturbing the execution of current applications and possibly creating new vulnerabilities.
- *Complexity.* Again for commercial reasons, systems are built with more functions that are needed for most applications. These functions cannot be removed and a user is forced to buy the whole package. The configuration of the many units may be complex also. This is unnecessary complexity but sometimes systems are complex by

nature. The Internet contains currently over 1 billion pages, a large variety of systems and users, it reaches almost every country in the world. It is also in constant change.

- *Outsourcing*. To reduce costs many companies outsource code development and maintenance to other companies, usually in countries with lower salaries. Errors can be introduced this way by the need to produce cheap and fast code, by communication problems, and maybe intentionally.
- *Myths and misconceptions about security*. There is lack of understanding of security in industry. Most software developers and managers know very little about security. Typical myths include blaming the user and administrators, believing that all attackers are outside, believing that cryptography and firewalls are enough, believing that "it cannot happen to us", and others (See Section 1.2)..
- *Variety of methods of access*—Users can access systems from mainframes, PCs, cell phones, TV set top boxes, pagers. There is a fertile ground for hackers to think out new attacks. The possibility of downloading programs or other contents is very useful but it brings new dangers. A downloaded program could contain a virus, damage files, etc.
- *New ways of working*—These include cooperative work, Internet access to medical files, online voting, and many others. Each type of application has its own security requirements and there are no policies or systems appropriate for all of them, which mean that we need to rethink them for each case.
- *Lack of company awareness and commitment*—This is the "It only happens to others" syndrome. There is reluctance to invest money and efforts in aspects that do not produce money. It is also hard to find trained personnel. In particular many new companies don't have the resources or knowledge to secure their sites.
- *It is hard to prosecute criminals*—Obsolete laws and lack of technical knowledge of most prosecutors makes it hard to convict a criminal. Hackers may become popular heroes, admired and emulated by many. However, this situation is slowly changing.
- *Uniformity of operating systems, utilities, and protocols*—Almost all operating systems used in the Internet are either Microsoft Windows, some variety of Unix, or Linux. The Internet uses only the HTTP protocol running over TCP/IP. Very few mail systems are in currently in use. This makes it easier for hackers to plan attacks or for viruses to propagate. This uniformity also makes the payoff bigger for the attacker. On the other hand, uniformity can be good in that patches and updates can be produced more frequently and training becomes easier.

1.5 The security environment

Although the whole history of computers is relatively short, computer system architectures have evolved significantly through the years and we have had at least the following stages:

- Centralized mainframes isolated from direct access from its users. Typically single-user systems. Almost no security problems existed in this environment.
- Mainframes where users can execute batch and interactive programs from terminals. Security problems of users attacking the database or other user spaces start appearing and becoming serious.

- Distributed systems with Personal Computers (PCs) and servers, connected together by an Intranet. More variety of attacks and more advanced attacks. Attacks from disgruntled or greedy employees.
- Distributed systems accessible also by external users through the Internet. The users of a system can access other systems. Unknown users, bidirectional danger, no geographic containment. Initially teenagers, later organized criminals.
- Distributed systems accessible from Intranets, from the Internet, and from mobile and wireless devices. New forms of software include web services, agents, and sensors. They all bring new forms of attacks.

Each architecture has brought new user advantages but also new security problems. In particular, the Internet was an enormous advance in permitting access to a large variety of data and in enabling a large number of activities to have a global reach. Its growth has been explosive and new uses for it are being found every day. In this moment one of its most serious problems is its lack of security, some high-profile incidents have made users wary and perhaps delayed or restricted their use of the Internet. The Internet has brought upon a fusion of internal and external systems, and its security problems cannot be considered in isolation of the involved internal systems. The current trends of web services and mobile devices add another dimension to the security problem.

Initially the only ones concerned with security were institutions having large databases. When PCs started to be networked, security became a concern of most users. The pervasive user of computers has given hackers a lot more possibilities of attack and more motivation. The increase in the complexity of systems has made it harder to defend them. Many old security problems have reappeared in different contexts.

In addition to web services and agents (discussed in Section 1.6) there is a variety of significant changes in the context for security. A variety of new devices or approaches have appeared, introducing significant new functionality. However, they also bring new security problems. Other changes are in the way of using information or building systems. We enumerate below some of the most important and we will discuss them in detail in later chapters.

Portable devices
These include cellular phones, PDAs, laptops, even iPods. They are usually wireless and in number they surpass by far the traditional computer systems. All these devices can be sources of vulnerabilities and malware specially oriented to them has started to appear. Their loss or theft provides a new source of attacks. The wireless medium in itself provides new ways to attack the systems. Because their location can be tracked there are also new privacy concerns. New standards such as WiMAX will make the use of these devices even more widespread.

Embedded systems
An embedded system is a computer system which is part of a larger system to which it adds some intelligence. Examples are computers in cars, cell phones, airplanes, washing

machines, and voltage controllers in power transmission lines. They introduce new attacks such as attacks that disrupt system timing [Koo04].

Sensors
Sensors can be used to detect fires, flooding, burglaries, and many other events. Some use RFID devices, others are more elaborate. They introduce many useful functions but also new security problems such as repeated interrogations to provoke battery drainage. RFIDs are low-cost sensors. They are very useful to keep track of inventory and locate physical items and many companies are starting to use them. They can also be used for automatic tolls in highways and in implanted medical devices. They bring up, however, serious privacy and other security issues.

Peer-To-Peer systems (P2P)
P2P systems are based on the idea of systems without predefined roles that send or receive information from each other. They use bidirectional web services, grid computing, or similar approaches. In a P2P system all the nodes are able to find and consume, or offering and fulfilling information and processing services.

Cloud computing
Pools of virtual resources can be shared by users who pay for what they use. This is an attractive platform for small institutions with variable number of customers, which don't need to make heavy investments to support their growth. Because the cloud is accessed through the Internet the usual security problems are present.

Cyber-physical systems
A cyber-physical system (CPS) integrates computing and communication capabilities with the monitoring and control of entities in the physical world. Such control systems are normally distributed, real-time, and usually include embedded devices, sensors, and wireless links

The Internet of Things (IoT)
The interconnection through the Internet of uniquely identifiable and addressable embedded devices. A thing can be a washing machine, a person with a heart monitor, a farm animal with a biochip, an automobile that has built-in sensors to alert the driver when tire pressure is low -- or any other natural or man-made object that can be assigned an IP address and provided with the ability to transfer data over a network.

Machine to Machine Communication (M2M)
Technologies that allow both wireless and wired systems to communicate with other devices of the same type

Other changes concern the general environment or the way to develop software:

Open source software

Open-source software is making inroads in areas such as operating systems, web servers, and compilers. Their relative security is a controversial aspect, some believe they are more secure, others consider them less secure [Cow03] (See exercise 1.4).

Outsourcing of software development
Many companies send the development, maintenance, or servicing of their products to countries with lower salaries. This practice introduces or exacerbates security problems, such as the possible introduction of malware in the software (see Exercise 1.3).

Computerization and exchange of medical information
Medical information is one of the most sensitive types of information. Its misuse could have a very serious effect on an individual's life; leakage of information about a psychiatric treatment could ruin a career, an incorrect change in a medical record may result in a wrong prescription with damage to the patient. In past times this information was collected and stored at physicians' offices and hospitals and relatively few people even knew it existed. In most instances it was not computerized and was protected by its isolation and the ignorance of its existence. All this is fast changing, most or all of the doctor offices use computers, hospitals have large information systems, and a good part of this information is becoming accessible through distributed systems, including the Internet. Specialized databases containing information about prescriptions or treatments are being linked together through the Internet. This means that the number of people that can potentially access information about patients has increased by orders of magnitude. This increases dramatically the potential for misuse.

Social networks
Social networks provide facilities for collaboration between people who have some common interest or belong to a similar group. A social network service includes representations of each user (a profile), his social links, and a variety of additional services. These networks are characterized by constant activity and their users put in them large amounts of personal information, which bring concerns about privacy. Already several security attacks have been reported in Facebook and Twitter, the two most popular Internet social networks.

1.6 The Internet and the WWW

Al abrir los ojos, vi el Aleph. El lugar donde estan, sin confundirse, todos los lugares del orbe, vistos desde todos los angulos. (When I opened my eyes, I saw the Aleph. That place where, without confusion, all the places of the world can be seen from every possible angle)
Jorge Luis Borges, "Narraciones", Salvat, 1982.

We look now at the structure of the Internet to define the computing environment that provides the context for attacks. We then analyze possible security threats. A system with access to the Internet is a high risk system because it may receive attacks from internal and external users and we can use it as paradigm. The Internet started as a network for academic collaboration, supported by the Defense Research Projects Agency (DARPA).

It was used initially mostly for e-mail and file transfer. Later, news groups and bulletin boards appeared. The World Wide Web (WWW) was a breakthrough that changed it forever. The WWW is a virtual network on top of the Internet protocol. The Internet and the WWW have increased their scope by orders of magnitude. The WWW is used almost everywhere in the world and it contains over a billion pages. It is close to fulfilling the idea of the Borges' story.

The Internet is a type of distributed client/server architecture. A *distributed architecture* is composed of autonomous computational units that interact (interoperate) with each other through messages and are connected through some type of network. Most practical information and control systems are distributed systems. Examples: the Internet, intranets in companies, wireless networks, aircraft control systems, distributed databases, and distributed file servers. The Internet uses several hierarchical layers, seen in Chapter 7.
The World Wide Web allowed the Internet to become a common communication vehicle and when it started it used only three basic concepts:

- Universal Resource Locators (URLs) for addressing and linking.
- Hypertext Markup Language (HTML) for information layout.
- Hypertext Transfer Protocol (HTTP) for information request and transfer (client/server model)

The basic architecture of the web (Figure 1.1) requires browsers to request HTML documents and provide URL caching. Web servers receive user requests, find and return documents. Documents are composed of pages and stored in files or database management systems (DBMSs).

Potentially all institution data can be considered documents. Web documents are hypertext /multimedia documents and can be passive (just display information) or active (contain links to programs). They can also be fixed (always bring the same information) or dynamic (assembled on request, see Figure 1.2). In this sense, Figure 1.2 is the high-level view of any information stored in computer systems and will be used as a guideline for information security.

Now, the trend is for documents to use XML.
XML is a metalanguage to define the meaning and structure of documents. XML is a subset of SGML (Standard Generalized Markup Language). XML uses tags in data items to define their meaning and can relate data items through nesting and references. We describe XML and its use in more detail in Chapter 8.

An important system development is the use of *components*. There is no consensus in their definition. In some views, a component is a unit of independent deployment, a unit of third party composition, and has no persistent state. Other definitions add contractually specified interfaces. Others say it must have state. They require supporting frameworks. Examples include: J2EE, COM+, CORBA, .NET. A *Web Service* is a type of component that is available on the web and can be incorporated in applications or used as a standalone service. Web services require a standard supporting framework, including

XML and a communications protocol. They provide valuable new functions but they bring in turn new security problems.

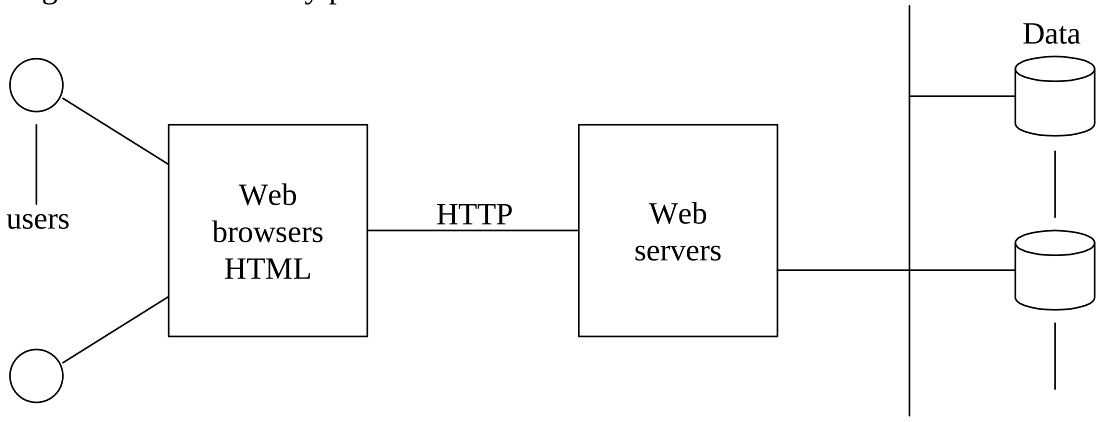

Figure 1.1 Basic Internet architecture

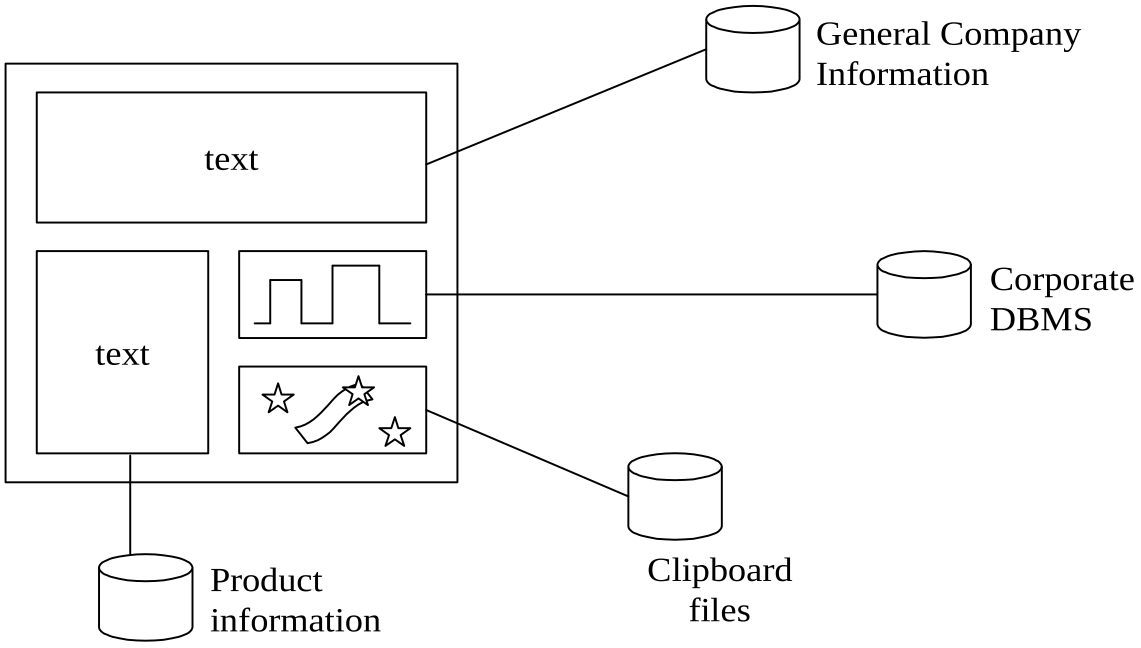

Figure 1.2 Structure of a dynamic page.

Agents are autonomous programs that move through the Internet performing specific actions; for example, looking for rare books, comparing their prices, and possibly buying some. They also bring interesting new security issues, e.g., they can be attacked by their hosts.

To host components and web services and to integrate systems, a new type of unit has emerged, the *Web Application Server*. A WAS contains a common model of the enterprise information and centralizes access to this information. It may also include identity management and a web portal. Figure 1.3 now becomes the architecture shown in Figure 1.4.

Figure 1.3 Internet architectures as seen from an institution

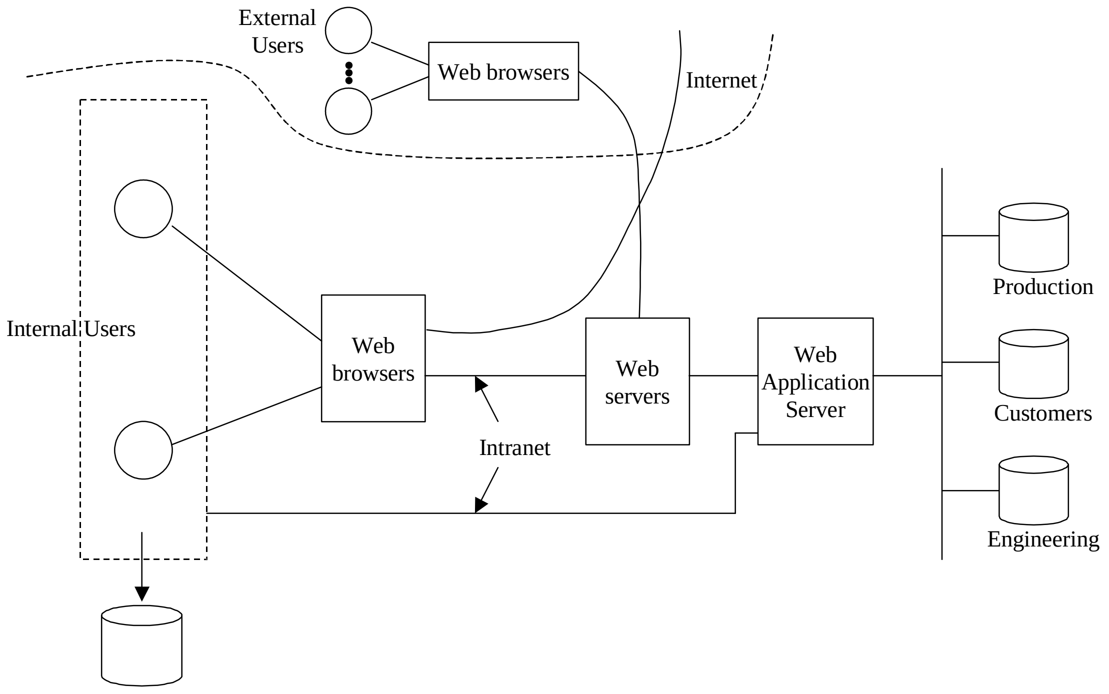

Figure 1.4 A typical institution architecture

1.7 Vulnerabilities and attacks

Before studying possible defenses we need to know the types of attacks that we have to stop and why do they happen. A *vulnerability* is a state or condition that may be exploited by an attack. People study source code and find flaws that can lead to possible attacks (*threats)*. An *attack* is an attempt to take advantage of a vulnerability and it may result in a *misuse*, a violation of security. An actual attack is an *incident*. The outcome of misuse can be loss of confidentiality or integrity, theft of services, or denial of service. An attack has a *perpetrator*, who has a *motivation (goal)* and probably appropriate skill. This motivation may be monetary gain (lucre), need to show off his skills, or to political/religious motivation. The attack has a *method of operation* (modus operandi) to accomplish a *mission* with respect to a *target.* The *damage* of a mission can be loss of assets, money, or even lives.

Vulnerabilities can be the result of design or implementation errors, although not every program error is a vulnerability. Design vulnerabilities are very hard to correct (one needs to redesign and reimplement the system), while implementation vulnerabilities can be corrected through *patches*. Examples of wrong design include using inadequate policies, lack of checks, lack of modularity (we'll see these in detail later). Examples of bad implementation include not checking size bounds for data structures and giving too much privilege to a process. Deployment, configuration, administration, and user errors are also sources of vulnerabilities. Vulnerabilities have a lifecycle [Arb00]. First they are discovered, then they are disclosed; the attacks increase until a patch is released. After that the attacks decrease. A good catalog of approaches used by attackers can be found in

[Bou98]. A very readable account of early attacks (still used currently by attackers) is in [Whi77].

Most of the *attackers* can be reduced to a few categories:

- Insiders—These could be motivated by lucre or be disgruntled employees.
- Malicious hackers (crackers)—This group includes vandals, exhibitionists, political activists, terrorists, and career criminals. These are usually outsiders.
- Spies—Industrial and government espionage are significant in some countries and some areas, e.g., semiconductors. These can be insiders or hackers.
- Organized crime—Usually, their motivation is monetary gain.

The basic steps of an attack are:

- Preparation (reconnaissance)—During this stage the hacker may do information gathering, scanning, planting malicious code, or masquerading (spoofing). A variety of tools are available in the Internet for this purpose [Bou98].
- Activation—It could be perpetrator controlled, timed, or victim activated.
- Mission—Active (affects integrity and availability), and passive misuse (eavesdropping, inference), denial of service.
- Propagation—This action may include possible infection of the new target.

The sequence diagram of Figure 1.5 shows the interactions for these steps. Not all attacks read or modify information or propagate. Some attacks infect the host (not shown in the figure).

1.8 Direct attacks and misuses

In a direct attack, the attacker takes advantage of a code flaw or configuration error to gain access to information. A typical attack is SQL injection, where the attacker inserts a database query in an input form. When we study the different architectural levels of the system we will show how specific attacks happen, e.g. SQL injection attacks are described in the chapter on databases. Direct attacks often are prepared through malicious code.

The objective of an attack is a misuse of the information to reach some goal for the attacker. An external attacker may try to find a way to transfer the list of customers of an institution; an insider can take advantage of incomplete authorization to modify some patient's prescription; an external attacker can infiltrate many computers and convert them into zombies that will be used later for a denial of service attack; a legitimate customer of a financial institution may deny having given an order to buy some type of stock.

1.9 Malicious code

A variety of programs have been written with the purpose of preparing or performing a misuse. They are what is called *malicious software or malware*. Catalogs and tutorials can be found in [bit, Bug, cer, Coh94, Gry05, ics, Sym]. Details of how they operate will be discussed in the chapters on the architectural levels where they operate, e.g. in the operating system. There are three basic types of malicious software:

- A *worm* is a program that propagates itself, self-replicates, may cause some damage, may drop programs to make the infected site a zombie, or drop a Trojan Horse.

- A *virus* is a program that attaches itself to another program (infects it), self-replicates, propagates, and usually causes some data or system destruction. A virus has three stages: 1) it is activated, 2) it infects a program, and 3) it carries out its mission when the program it has infected is executed. A virus has a *payload* that may do something harmful.

- *A Trojan Horse* is an apparently useful program that has hidden functions (*covert* effect), usually harmful. They may also include trapdoors (see below).

Other varieties of malicious code include:

- A *logic bomb* is a program executed when a trigger condition becomes true.

- A *bacterium* or *rabbit* [Bis03], is a program that takes over all or some class of resources.

- *Spyware* (snoopware) are programs that watch user behavior such as keyboard actions in order to collect passwords, credit card numbers, and similar. Some collect information about a user's habits and interests with the purpose of targeting advertisements and are called a*dware*. Some appear to perform a useful function, e.g., a calendar, and are then special cases of Trojan Horses.

- *Spam* is the unsolicited sending of many copies of a message, usually intended to sell something. It is not strictly a security problem but it can be considered a violation of the privacy of the recipients. It can also be used to carry other types of malware such as viruses. Several approaches have been developed to filter unwanted messages [Vau03]. The real problem is the lack of effective legislation to control spammers.

- A *phishing message* misrepresents itself to get information from users. For example, the user receives a message supposedly from his bank asking her to enter her account information for control purposes.

- A *bot* or *zombie* is a computer that can be controlled by another through the introduction of a Trojan Horse or another type of malware.

- *Rootkit:* Any software that takes control of an operating system, allowing illegal administrative functions. They alter parts of the operating system kernel and hide their presence.

- *Advanced persistent threat (APT)* is an attack from an organized group, such as a government-sponsored group, that performs a sustained and organized attack against a specific target, usually with intent of espionage.

- A *remote administration tool (RAT)* is software that allows a remote operator to control a system as if he has physical access to that system. While it has many legal uses, "RAT" software is usually associated with criminal or malicious activity. Malicious RAT software is typically installed without the victim's knowledge, often as payload of a Trojan Horse, and tries to hide its operation from the victim and from security software.

Some malware are combinations of the basic types and can combine worms with spam or spyware, these are called *blended threats*. During an attack, the malware may do some actions to hide its presence. A *stealth* virus/worm hides its presence by using encryption or wrapping itself in another file, possibly compressed. . Some attacks try to destroy evidence by erasing logs and other tracks of intrusions. A *polymorphic* virus/worm modifies itself when propagating. *Metamorphic* malware uses code obfuscation approaches such as code transposition, module rearrangement, and extra code, to produce variants of a core/base malware that are thus harder to detect. The variants have the same functionality as the core but different signatures, which makes them harder to identify by a signature-based detector. A *zero-day attack* is an attack that takes advantage of a vulnerability just discovered (maybe only by the hacker) and for which there is no patch or detection method yet. Examples of zero-day attacks are the WMF and VML vulnerabilities in Internet Explorer. *Tunneling* is an attack below the level of some defense. Some malware leave trapdoors. A *trapdoor* is a section in a program that allows an attacker to get into the system at a later time; for example creating an additional entry with a known password. In some viruses only a small piece of code is propagated, once installed in a new system it download more code from some network site. Some attacks are directed specifically to some defenses: Authentication attacks target passwords or other authentication methods. Disabling of links to patch centers and subversion of application controls are other approaches.

General malicious program detection is an undecidable problem (equivalent to the halting problem in Turing machines) but we can detect known types of viruses or worms. There are several companies that have products for this purpose: McAfee, Finjan, Symantec, TrendMicro, Norton, and others. Safeguards include: prevention (safe computing), proper configuration, detection (antiviral software), identification and containment, and recovery (keeping back up copies of the data).

Malicious software invades a system in two basic ways: a user opening an attachment with executable code, or a user visiting a hostile web site. A third possibility is direct injection through an unprotected port. Downloading infected files from the Internet or reading files from flash memories can also bring a virus infection.

Some notorious viruses/worms have been SoBig, Nimda, Anna Kournikova, Hybris, I Love You, Melissa, Blast, Bagle, and Sasser. Each one of these has many variants. A typical modus operandi is the one in Hybris, a worm that propagates by monitoring a PC's network connection for messages. When a message comes, the addresses in its header are added to a list. The list is used later to send copies of the worm. It also changes shape (polymorphic behavior) by adding extensions to its code, retrieved from some Internet addresses. Usually, to get infected one must open an attachment containing the virus; however, some viruses can penetrate taking advantage of open ports and other configuration errors.

The number of vulnerabilities found in complex systems keeps increasing; a recent survey by Symantec shows a sharp jump in Windows-based systems giving an average of 48 new vulnerabilities a week. CERT statistics show 5990 vulnerabilities in 2005 and 8,064 in 2006. At the same time, the speed of propagation of viruses keeps increasing. Nimda took 22 minutes to become number one (in reported attacks), a few years ago it took 'Friday the 13th' about three years to reach this status. There is also concern about their improved sophistication and virulence [Che04]. According to some opinions, organized crime is becoming more involved in computer crime. On the positive side, worms may be useful and have been used to detect system failures and perform maintenance in distributed systems [Sho82]. Similar technology could also be used to implement agents.

Jnanabot is an example of a bot that infects Windows, MacOS, and Linux. It spreads by posting links in a victim's social-networking accounts and sending attacks based on friend lists. The initial platform-independent Java script downloads platform-specific components to complete the infection

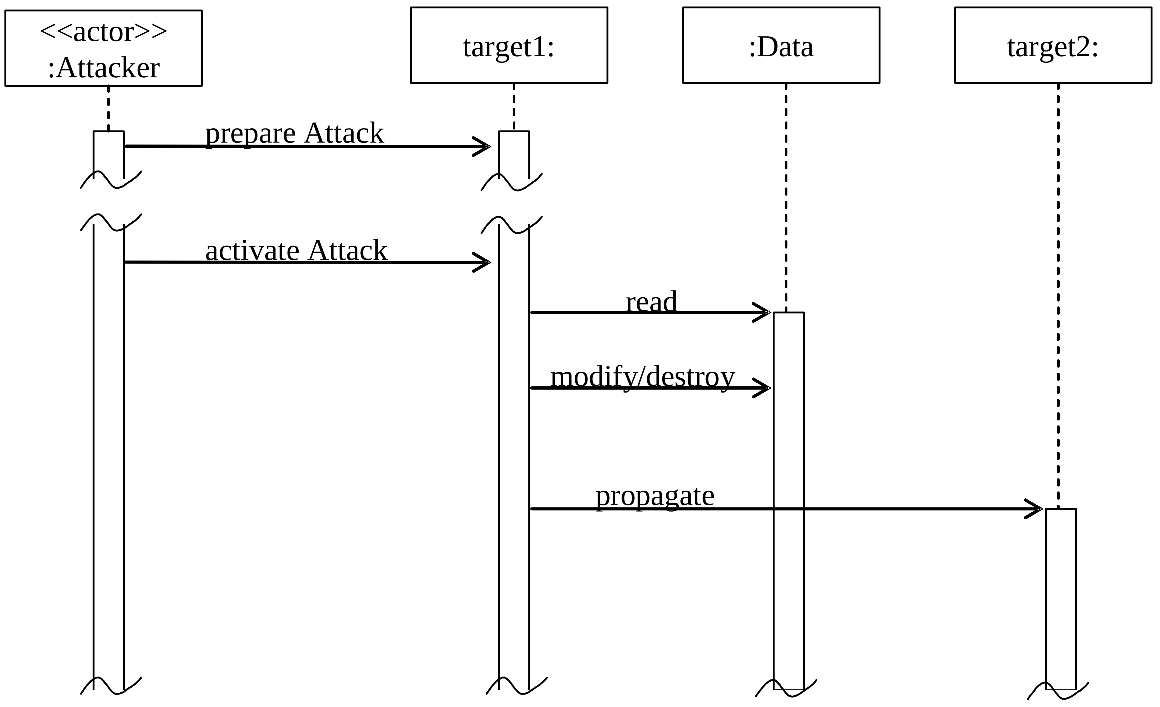

Figure 1.5 Sequence diagram for the steps of an attack

Until recently, hacking involved little risk and possible rewards. Hackers sometimes even became local heroes. This is slowly changing:
--In November of 2003 Microsoft started a $5M reward program to encourage people to turn in or catch hackers.
--In January of 2004 Helen Carr was sentenced to 46 months in prison for using forger e-mails supposedly from AOL security to coerce AOL subscribers to release their credit card numbers (an example of phishing).
--The British Terrorism Act 2000 considers as terrorists "people who endanger lives through the manipulation of public computer systems".
--An extreme case happened in China in 1999 when a hacker (Fang Yong), who stole money from a state bank, was condemned to death (apparently changed to life in prison later). He had hacked the computer network of the Industry and Commerce Bank of China and shifted about $87K into his account.

1.10 Examples of incidents

Some early (before the Internet) attacks can be found in [Neu95]. Skoudis has a history of malware as well as many details on specific incidents [Sko04], while Symantec keeps a catalog of malware [Sym]. Typical early cases include:

- The "salami slice" approach, where the attacker takes small amounts of money from many accounts [Whi77].
- The manipulation of insurance policies, where policies of fictitious or dead people are reinsured with the purpose of fraud. Here the computer is used to make the policies and their behavior look authentic [Whi77].
- A programmer in a company in France was fired in 1971 [Per84]. He was in charge of writing file update utilities. Two years later one of these programs wiped out all the files of the company.
- Embezzlement. People have used computers to hide misuse of funds to which they had access.
- Data "diddling". An auction company in Sweden in 1979 accepted written bids on goods, where the high bidder paid the amount of the second highest bid plus 10 percent [Per84]. The company would get 8 percent of the sale price as commission. Company members used a computer to insert false second highest bids to inflate their commissions.

More recent incidents include:

- A hacker in the UK in the year 2000 got the names, addresses, passwords, and credit card numbers of over 24,000 people from an Internet service provider.
- Hackers forced the Hong Kong government's website to shut down in 2001 by defacing its web pages.
- A hacker in South Korea defaced over 720 web sites in 2000.

- A group of hackers from several countries threaten to attack the network of the Indonesian government, army, and financial institutions in 1999 to create chaos in case they would not grant full independence to East Timor.
- Mydoom (or Novarg), is a worm that appeared in January of 2004. It infected about 100,000 computers in 142 countries and became one out of 17 messages in the Internet (20% of the traffic). This virus installed a backdoor that allows a hacker access to the hard drive. It then searches files looking for addresses where to send itself. Its main purpose was to send a flood of messages to the SCO Group, involved in a dispute with Linux vendors.
- A man in New York City planted key logging spyware on public computers at 13 Kinko's copier stores and stole private information of about 450 people. He then used this information to get money from their bank accounts.
- In 1996, Sky Dayton, the chairman of EarthLink, an Internet service provider, was defamed by postings that indicated that the company was controlled by the Church of Scientology.
- During February of 2000 coordinated denial of service attacks blocked access for several hours to Amazon, Yahoo, Buy.com, CNN, eBay, E*Trade, and ZDNet.
- Three people in Miami were recently indicted for the theft of 130 million credit card numbers [Sto09].
- A virus invaded Facebook in December 2009 and sent many messages offering merchandise in the name os some of the members.
- In late 2009 a malicious program attacked Nokia phones and made small charges to the owner's account. Phones can do more and more functions and with them come a variety of possible attacks.
- A massive attack against Google was performed by Chinese hackers in January 2010, attempting to compromise the accounts of government opposition activists.

As a measure of the current state of the Internet we enumerate the most frequent types of attacks that have occurred in the Internet [Fer01]. In our enumeration, we mention the behavior of each attack, its effect, a possible motivation for the perpetrators, and the way the attack works (see below). Most attacks can be classified into a few groups according to their objectives or type of misuse:

Viruses and worms. As discussed earlier, they may destroy information, clog services, or drop spam or spyware. Initially, a mix of vandalism and ego trip drove typically young hackers to create them but now they are created mostly by organized groups with monetary objectives. To propagate they take advantage of operating system flaws and the uniformity of Internet platforms [Bou98].

Web site defacing and hijacking. This attack involves alteration of the web pages of some institution, destroying or replacing them. Visitors may be hijacked to other sites, usually pornographic sites. Another related attack is for a site to simulate another to get client information. Usually political motivation is behind this type of attack, a group protesting for some immediate reason or some general principle, although economic gains are usually the reason for impostor sites. These attacks take advantage of web server weaknesses.

Illegal access to information. Usually involves access to databases or files connected to web servers and implies stealing of information, e.g., credit card numbers. This attack has economic reasons or blackmail. It exploits poor database authorization or alternate routes to the database that may bypass this authorization.

Privacy violations. Unauthorized collection of user information for economic or defamation purposes. Can be a variation of the previous case but can also be performed through cookies and monitoring programs (spyware or adware).

Identity-related attacks. The attacker using phishing, spyware, or information posted on the web prepares a profile of a person, assumes the identity of that person and opens credit card or bank accounts that are then used to acquire goods or spend money in other ways. Identity-related attacks have become the most common types of attacks [Pag07].

Distributed denial of service. It is produced by a proliferation of messages from several sites towards another. A site may become inaccessible due to the large volume of messages. Usually done by political motivation or vandalism. It requires previously inserted software by the perpetrator in the slave sites or inadvertently downloaded by the slave. Takes advantage of network protocols, operating system resource policies, poor control of remote operations, and lack of security in many PCs connected to the Internet.

Cyberterrorism or cyberwar. This implies a massive attack to the computer systems of some country. Because it requires a large amount of resources, it can only be carried out by organized groups or by a government. In some cases it may become a cyberwar and it may use combinations of all the other attacks. An alarming aspect is that there have been recent attacks against a train signaling and dispatching system, a nuclear power plant, and a hospital, all of which are considered safety-critical systems. Another possibility is attacks against the structure of the Internet itself.

1.11 The design and evaluation of secure systems

There is no 100% security. In fact, we don’t even have quantitative measures of security. We can only define qualitative measures; for example, by demonstrating that a system can handle its most likely threats. We discuss that in detail later. We consider now a possible approach to build secure systems.

This approach is based on modeling and uses patterns (see Section 1.13) and we can show that a system built according to this approach can stop or mitigate specific threats. The coding stage can introduce new vulnerabilities and if we want a very secure system we need to do code checking. Microsoft’s SDLC is an example of an approach based on systematic code building and checking [How06]. The use of models can lead to systems where whole ranges of threats are not possible and can reduce the amount of code checking. Another advantage of modeling is that, even if the attack still happens, its effects can be better controlled; for example, a virus cannot propagate. A design error requires redoing parts of the system and may be very costly to repair; code errors are corrected by patching, which do not require extensive changes. As we will see in Chapter 4, the authorization system of Unix has design and coding errors. The latter have been corrected in later versions while its design errors remain or need a much more complex fixing.

To understand better the structure of complex systems it is common building them as a hierarchy of layers. This approach has several other advantages as we will discuss later. It has even been defined as a pattern (see Section 1.13). We can think of a computer system as consisting of several hierarchical layers that go from user applications to the hardware. We can associate the attacks discussed earlier with the architectural layers of a system. Figure 1.6 shows the typical attacks that may occur in each layer. At the highest layer we have the software architecture that defines the program structure (classes, associations, modules); this can be considered a metalayer. A wrong design could be a source of attacks. An attacker can also infer information by looking at a design model. Executing applications (the next lower layer) can be attacked through wrong operations, malformed (too long, wrong type) parameters in operations, attacks through the language (incorrect use of pointers), unauthorized data access because of lack of controls, viruses and worms (See Chapter 5). The database management system (DBMS) layer can be attacked through malformed parameters in queries, lack of authorization, and other ways (See Chapter 6 for a discussion). The operating system (OS) layer is subject to directory attacks, interference between executing programs, root directory attacks (superuser mode), virtual address space violations, unauthorized file access, and others (See Chapter 4 for a discussion). The hardware and communications layer is subject to physical attacks (removal of disks), physical eavesdropping and other network attacks (See Chapter 7).

The fact that attacks can happen in any layer requires that a security expert knows the complete structure of the system. While nobody can be an expert in all the layers, a good knowledge of their functions and requirements is important to understand which defenses are needed and where.

One of our main purposes is to show how to build secure systems and we will discuss design aspects throughout the book. For each layer we will consider its possible attacks and how to defend against them. Some fundamental principles, obtained from experience and system theory, to design or assemble secure systems include:

- *Complete lifecycle*. Security must be applied throughout the whole development lifecycle. Security must be considered from the inception stage of a system, adding it later is not a viable approach.
- *Risk analysis*. Enumerate threats as attacker goals and evaluate their possible effects. Before starting development, we must assess risk and decide how much security we need to put in a system. Risk analysis identifies important assets of the system that need protection and the effect of their compromise. Probabilities of occurrence and the cost of attacks are hard to evaluate but onloy approximate values are needed.
- *Functionality*. Define security mechanisms side by side with functional requirements along the whole lifecycle..
- *Policies*. Use an explicit and consistent set of policies to guide the design of the system. Security *policies* are high-level guidelines for security (see Chapter 2). Each threat should be controlled through some policy.

- Select *mechanisms* to implement these policies using combinations of software and hardware. In each architectural layer we need appropriate mechanisms, which can be described by patterns.
- *Usability.* The secure system must be easy to use by administrators and transparent to the users.
- *Overhead.* The security mechanisms should not add too much overhead, and should be flexible and extensible
- *Hierarchical structure.* Look at the system as a set of hierarchical architectural layers. All the levels of the system must be considered, security is a cross-section problem, protecting one level just invites attacking the system at a different level. These layers must include software and hardware layers.
- *Defense in depth*. Because we cannot be sure that we have covered every possible attack we use multiple and overlapping protection systems. The use of redundancy requires careful management or it will make things worse, not better. Too many security mechanisms will impact performance, cost, and complexity.
- *Trust*. Minimize the system units that have to be trusted.
- *Security administration* must be clear and systematic or the administrators will make mistakes that will result in security exposures.
- *Monitoring*. Once the system is deployed, monitor the system constantly. In this way we can add missing defenses or improve existing defenses.
- A good *design methodology* is important, we believe the object-oriented approach is far superior to other approaches to produce secure systems.

Clearly we need to define a context for these principles, learn when to apply them, and how to apply them. We define more specific principles in Chapter 2.

Most of the design of a secure system is software design, hardware is either given or selected from standard architectures. The way software is developed (software process) and the specific methodologies used for it are very important to produce secure systems. As indicated earlier, security must be a concern from the start of development.

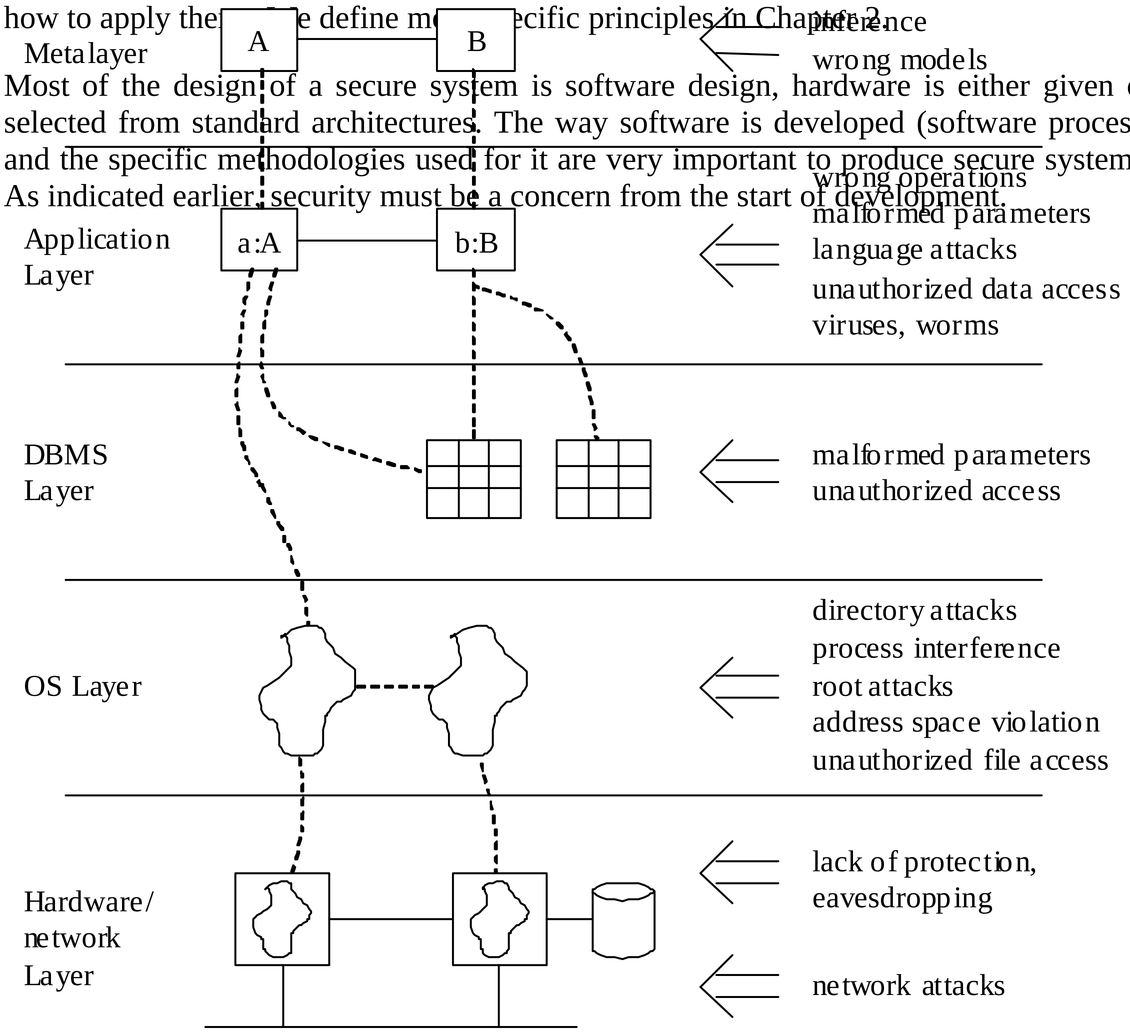

Figure 1.6 Some possible attacks in each architectural layer

As indicated, security principles should be applied at every development stage. We sketch now a secure software development cycle we consider necessary to build secure systems. We will use this methodology as a guideline throughtout the book. Figure 1.7 shows a secure software lifecycle, indicating where security can be applied and where we can audit for compliance with security policies

Requirements stage: *Use cases* for the system are defined at this stage, where a use case describes an interaction with the system. We can enumerate threats with respect to use cases. Security constraints can also be defined in use cases (secure use cases). Application-specific regulations should also be applied here.

Analysis stage: From the use cases we can determine the rights for each actor. We can build a conceptual model where we can apply patterns to realize the rights determined from use cases. Analysis patterns can be used to build the conceptual model in a more reliable and efficient way. In fact, analysis patterns can be built with predefined authorizations according to the roles in their use cases. This makes the job of defining rights even easier. This is discussed in detail in Chapter 2. We can verify that all the enumerated threats are controlled by some pattern.

Design stage: Interfaces can be used to enforce the security constraints defined in the analysis stage. Remember that user interfaces should correspond to use cases. Components can be secured by using rules defined according to the authorization needed for each component. Distribution may require additional security. Deployment diagrams can define secure configurations, to be used by security administrators. We can now verify that our mechanisms take care of all the types of attacks defined in the requirements stage. In order to select the appropriate mechanisms we need to look at how the attacks for each use case are actually realized through the architectural layers of the system.

Implementation stage: This stage requires reflecting in the code the security constraints defined in the design stage. Because these constraints are expressed as classes and associations, they can be implemented as the rest of the application. We may also need to select specific security packages, e.g., a firewall product, a cryptographic package.

At the end of each stage we can perform audits to verify that the institution policies are being followed and that the threats are being controlled. If necessary, the security constraints can be made more precise by using some type of formal constraints (See Chapter 2). A *domain model* can be built to support related applications; for example, Academic applications. In this model we can also apply regulations and institution policies as well as consider threats.

After we build a secure system we need ways to evaluate the degree or level of security reached. Until now we don’t have numeric measures of security. All our measures are qualitative, usually based on how the system has been built and on tests of the code. NIST has defined seven levels of security, where EAL 7 is the highest [nis]. The highest level of security reached until now by a commercial system is an operating system kernel for avionics, certified at the EAL 6+. This level implies formal verification and code testing. This kernel certification took about 10 years, it is not clear how applicable it is to more general systems Most commercial systems have new versions every 2-3 years and are quite dynamic. Having a systematic way to enumerate threats we can try to show that in the final product those threats are not possible, both from a design perspective (examining models), and from a code perspective (examining and testing the code). Formal proofs are only practical for relatively small and static systems.

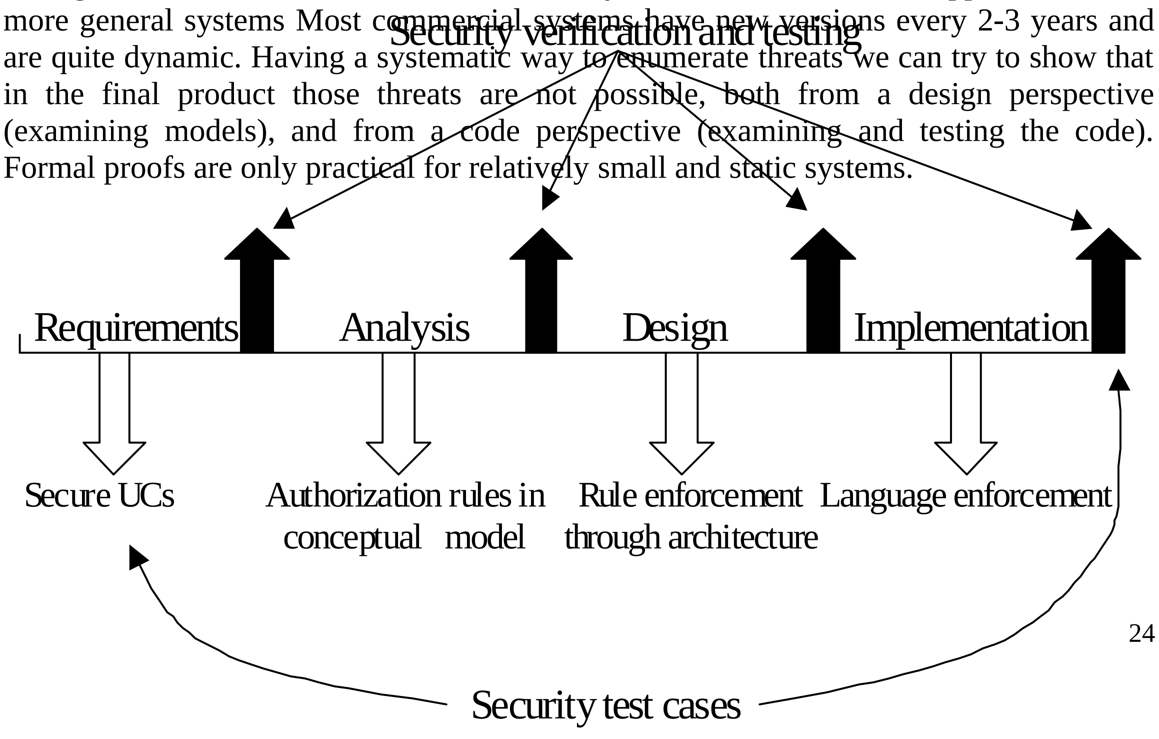
Security verification and testing
Requirements
Analysis
Design
Implementation
Secure UCs
Authorization rules in conceptual model
Rule enforcement through architecture
Language enforcement
Security test cases

Figure 1.7. Secure software lifecycle

1.12 Standards, consortiums, and public security services

There is a variety of standards for security, defined by organizations such as IEEE, Internet Engineering Task Force, and others [Mer03]. In 1983 the U.S. Dept. of Defense published a document that was used for many years to evaluate computer security. This was the “Orange book” [DoD], that provided a set of categories for classifying the level of security of computer systems. Because of its military origin, that book emphasizes multilevel security and has lost relevance for commercial systems. The NIST has introduced the Common Criteria for evaluating a larger variety of systems and it is now the most used standard [nis]. We will discuss these standards later in more detail. There are also more specific standards, e.g., for web services security; these will be discussed in the respective chapters.

There is a variety of groups seeking ways to improve the security of systems:
The Trusted Computing Group (TCG) (http://www.trustedcomputinggroup.org/) is an industry group that defines standards for software and hardware products [tcp]. TCG includes over 190 companies. Their first push is towards hardware-based security, a controversial approach (see Chapter 3).
The Center for Internet Security (CIS) has developed a set of security benchmarks for operating systems [cis].
The Forum of Incident Response and Security Teams (FIRST) is an international consortium of computer incident response and security teams who work together to handle and prevent incidents [fir].
The Information Technology-Information Sharing and Analysis Center (IT-ISAC) is an alliance of several companies to share information about product security [iti].
The *Security Industry Association* (www.siaonline.org/)*. Represents manufacturers of security products and services.*

For companies that do not have expertise on security there is the possibility of outsourcing. Several companies offer services to manage all the security aspects of another company. There are advantages and disadvantages to this approach but for small companies it is an attractive option (see Exercise 1.3). Another useful security service is offered by the U.S. Post Office and similar organizations in other countries, which have online secure services to transfer sensitive documents and provide digital certificates (see Chapter 3). .

1.13 Security patterns.

A unique feature of our treatment of security is the use of object-oriented models, in particular *patterns.* Patterns are one of the most important developments in software. A pattern embodies the knowledge and experience of software developers and can be reused in new applications. A pattern solves a specific problem in a given context and can be tailored to fit different situations. Analysis patterns can be used to build conceptual models, design patterns can be used to improve the software design, and security patterns can be used to build secure systems [Sch06]. *Security patterns* are represented here using the Unified Modeling Language (UML) [cet, OMG]. The object-oriented approach is semi-formal yet intuitive and is well known by most software developers. As first pattern in this book we show the Layers pattern. Typically, patterns are described using a template that we introduce later.

As indicated, security encompasses all the architectural levels of a system. The *Layers* architectural pattern [Bus96] is therefore the starting point of the design of secure systems. This pattern provides a structure where we can define patterns at all levels that together implement a secure system. Its main idea is the decomposition of a system into hierarchical layers of abstraction, where the higher levels use the services of the lower levels. Here it provides a way to put things in perspective and to describe the mechanisms needed at each layer. Figure 1.8 shows a typical set of layers used to analyze secure systems. This figure shows some of the participants at each layer and their correspondence across layers. Some of the layers shown can be further decomposed into sub-layers or the decomposition can be done along slightly different aspects, e.g., emphasizing the DBMS. An alternative decomposition can be useful for distributed systems [Uzu13]. We used this pattern earlier, in Figure 1.6, to show possible attacks that can happen at each layer. An important aspect of a pattern is the discussion of the advantages and disadvantages of the proposed solution (Consequences section). For example, among the advantages of the Layers pattern we have the flexibility to make changes to the lower levels without affecting the higher levels and the possibility of hiding sensitive aspects of the system. Among its disadvantages we have the extra overhead due to indirection.

1.15 Resources

To get a structured and systematic introduction to security, the simplest approach is to find a good book. This is not so easy. While there are many books about security, a good percentage of them are not very good. Most books fall into one of these two categories:

- General security textbooks, intended for general introduction to the subject. They try to cover most of the relevant topics. Some of the best are [And08, Bis03, Gol06, Pfl06, Sta08]. [Bis03] is a detailed book with lots of theory, while [Gol06] is concise and conceptual; [And08] has a descriptive, practical approach. A more recent book is [Sta11].

- Books on specialized topics. These may cover cryptography, network security, web services security, or similar topics. We'll mention some of those when we discuss the corresponding topic.

Another classification of security books is:

- Practitioner-oriented, with little or no theory and usually containing collections of examples. These do not provide a conceptual foundation but may be acceptable to study specific systems and for implementation details.
- Books oriented to university courses. Their technical level is generally higher and they require some computer science background. The books mentioned above fall in this category.

Web resources are useful because of their easy availability but their level of quality is variable. Some are sites with news about incidents or new technical developments, e.g., [cer, dev, goo, its, msd, nis, san, yah]. Others present vendor material, which is useful if you are looking for descriptions of specific systems, but you should not trust their evaluations or comparisons. The site of the National Institute of Standards and Technology (NIST) has many security publications [nis], including a collection of early computer security poapers. An interesting set of topics can be found in [Mar]. The University of the Aegean has a good set of links [aeg]. A problem with web sites is that their URLs may change.

A good number of security journals are published, e.g., [acm, cas, iee]. There is also a journal dedicated to privacy [jpt]. Some of the general technical journals publish articles about security, e.g. [act, iee, msd]. A few of these are available in the web [dev, sys]. Good sources of news are The New York Times, The Register (UK) [reg], and SANS Newsbytes [san]. Th ide mos s; for irreverent (and serious) opinions you should read Crypto-Gram, published by Bruce Schneier [Cou]. Another interesting place is Peter Neumann's web page [neu]. CERT [cer] and Bugtrack [Bug] publish lists of vulnerabilities and incidents. For p ve have EPIC [epi]. Linka about object-oriented design can be found in [cet].

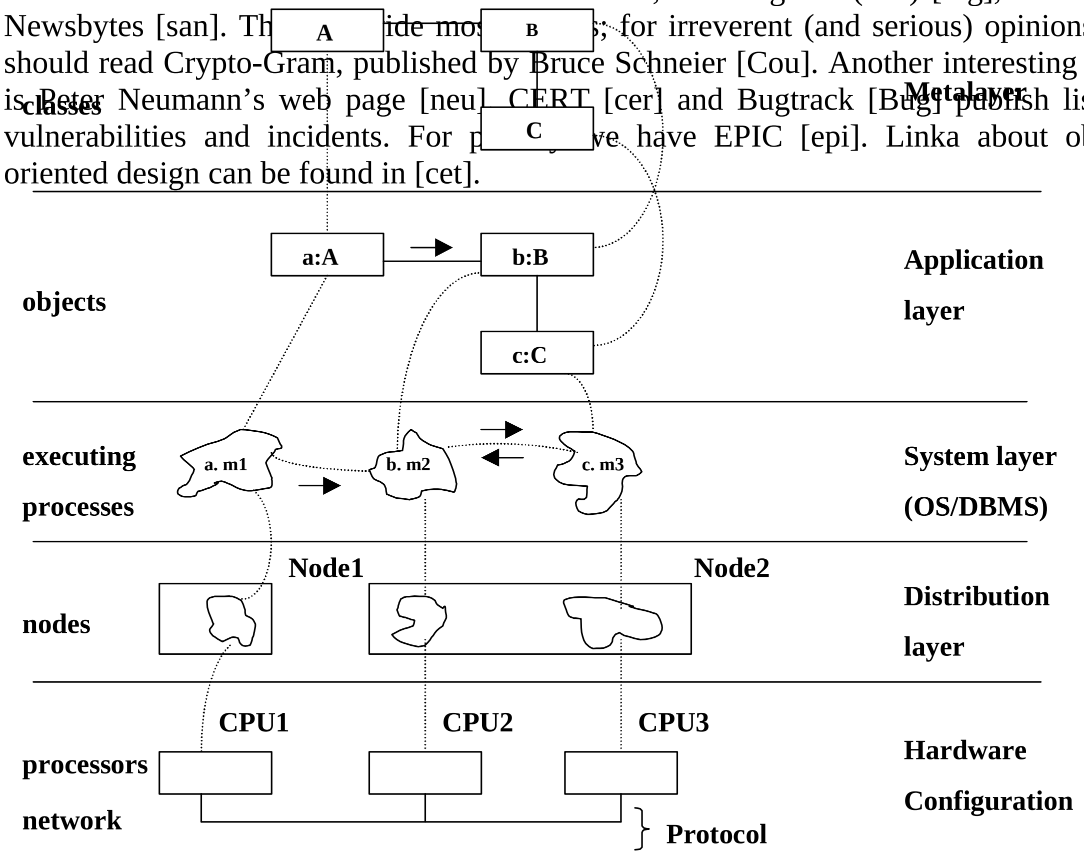

Figure 1.8 An instance of the Layers pattern.

Conferences are useful to attend tutorials on latest developments, to see new products, or to see the current research on security. There are several research conferences about security

IFIP WG 11.3 Working Conference on Data and Application Security:
http://seclab.dti.unimi.it/~ifip113/

IEEE Conference on Security and Privacy: http://www.ieee-security.org/

ACM Computers and Communications Conference, http://www.acm.org/sigsac/ccs.html

European Symposium on Research in Computer Security (ESORICS):
http://www.laas.fr/~esorics/esorics.html

Other conferences are industry oriented and emphasize current products and practical aspects:

RSA Data Security Conference: www.rsaconference.com/

Computer Security Institute Annual Computer Security Conference:
http://www.gocsi.com/#annual

SANS Annual Conference: http://www.sans.org/

Another possibility for a general introduction to security is to take a course. Many universities offer graduate or undergraduate courses. SANS [san] and the Computer Security Institute [csi] offer a variety of short courses. Some security consulting companies also offer courses.

1.16 This book

We cover here the most important aspects required to understand and to apply security principles. They are complemented with many references and links to other publications, as well as a good number of exercises. The three authors of this book are active security researchers and some of this material comes from their papers. Some sections have some contents from an earlier book of the first author [Fer81]. As indicated earlier, we make heavy use of UML because it can bring more precision to the discussion without being overly formal.

The first three chapters provide the basic conceptual structure of secure systems, the remainder of the chapters shows how these structures are implemented in different architectural layers. This means that the first three chapters should be read first while the other chapters can be read in arbitrary order. The chapters include:

Chapter 2.. Discusses policies and models used in current systems or of theoretical value. This chapter provides a conceptual structure for the specific mechanisms seen later.

Chapter 3. Cryptography. Evaluation and perspective of approaches to provide secure communication channels. Survey of most important approaches. Digital signatures. Authentication protocols for distributed systems.

Chapter 4. Considers operating systems and their relationship to hardware. We survey weaknesses in operating systems and how they have been corrected in some systems. The chapter also discusses Authentication.

Chapter 5. Program and application security, considering aspects of languages and application software architecture.

Chapter 6. Database systems, in particular relational and object-oriented databases. Discusses in detail access control aspects.

Chapter 7. Discusses network security, including wireless aspects. Secure protocols.

Chapter 8. Wireless systems, including smartphones.

Chapter 9. Internet and distributed systems, including web services and cloud computing. Cyber-physical systems.

Chapter 10. Developing secure software. Methodologies to develop secure systems. Evaluating security.

Exercises

1.1. A computer system doesn't perform authentication. However, it performs Authorization (assume it works properly). What kinds of general security attacks are possible here? Consider also the effect of having or not having logging. By general, we mean not tied to some implementation.

1. 2. Another system performs authentication but no authorization. What general attacks are possible now? Consider also the effect of having or not having logging/auditing.

1.3 Outsourcing software development to other countries with lower salaries has become popular as a way to reduce costs. Analyze the potential security problems which this practice may bring.

1.4 Some companies outsource their security functions to a company that selects and maintains security products, monitor attacks, perform scans for vulnerabilities, collect logs, and similar security functions. Discuss the advantages and disadvantages of this approach for a given company (e.g. yours).

1.5 Analyze the security advantages and disadvantages of open source software with respect to closed source software.

1.6 Draw a sequence diagram for the case of a Distributed Denial of Service attack.

1.7 Microsot has proposed a threat classification called STRIDE. Analyze it critically and compare it to the security objectives of Section 1.1

1.8 Collect recent Internet attacks and classify them according to the groups discussed In Section 1.8.

1.9 Analyze the two examples of malware below: What type of malware are they? At which architectural levels do they operate? Justify your answers.

a) The Bagle malware works as: if a user opens its executable attachment it copies itself in the system directory. Then it runs a program to download another part of it from a specific address in the web. Finally, it scans the hard drive searching for email addresses. With each address it finds it sends itself using its own SMTP (Simple Mail Transfer Protocol) engine (a mail server).

b) A hacker takes advantage of a flaw in an Internet database form (for example to order some product) to take control of the data management system. He then copies a file that contains customer credit cards.

1.10 Would you hire somebody convicted of some computer crime to be the security expert in your company? Justify your answer.

1.11 Databases are becoming more and more linked, e.g. medical, law, driving, real estate information is getting all connected. This means that it is rather easy to get all kinds of information about individuals. One individual said: "I don't mind, I have nothing to hide". What is wrong with this position?

1.12 The US Government has a ‘no-fly’ database, used in all airports, that contains the names of potential terrorists who should not be allowed to fly. Discuss the advantages and disadvantages of this approach.

1.13 What requirements should be imposed on a Logging service so that it has forensic value, i.e. it can be used as prosecution evidence in a court of law?

1.14 Do you think the government should force critical industries, such as electric grids, water treatment, air transportation, to have security measures in place?

1.15 Is a high level of Internet anonymity a good thing? Discuss its advantages and disadvantages.

References

[acm] Communications of the ACM, www.acm.org

[act] ACM Transactions on Security: www.acm.org

[adv] Security Advisor, http://securityadvisor.info

[aeg] http://www.icsd.aegean.gr/info%2Dsec%2Dlab/

The Infosec Lab of the University of the Aegean in Greece keeps a good set of links to security research groups:

[And08] R. Anderson, *Security Engineering (2nd. Ed.)*, Wiley, 2008.

[Arb00] W.A.Arbaugh, W.L.Fithen, and J. McHugh, "Windows of vulnerability: A case study analysis", *Computer*, December 2000, 52-59.

[Ber01] H. Berghel, "Cyberprivacy in the new millennium", *Computer*, IEEE, January 2001, 132-134.
Includes a good list of web resources about privacy.

[Bis03] M. Bishop, *Computer Security: Art and science*, Addison-Wesley 2003.

[Bis97] M. Bishop, S. Cheung, and C.Wee, "The threat from the net", *IEEE Spectrum*, August 1997, 56-63.

[bit] http://viewer.bitpipe.com/viewer/viewDocument.do?accessId=12123416

[Bou98] A.Boulanger, "Catapults and grappling hooks : The tools and techniques of Information warfare", *IBM Sys. Journal*, vol. 37, No 1, 1998, 106-114.

[Bug] Bugtraq archives, http://www.securityfocus.com/archive/1

[Bus96] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerland, and M. Stal., *Pattern-oriented software architecture*, Wiley 1996.

[cas] Computers and Security, www.sciencedirect.com/science/journal/01674048

[cer] CERT Coordination Center, http://www.cert.org/

[Cet] Cetus patterns links, http://www.cetus-links.org/oo_patterns.html
(this is a set of links about object-oriented systems)

[cexx] Cexx, http://www.cexx.org/adware.htm

[Che04] B. Cheswick, "Viruses becoming more complex and vicious", *Computer World*, 2004,http://www.computerworld.com/securitytopics/security/virus/story/0,10801,93444,00.html

[cis] Center for Internet Security, http://www.cisecurity.org

[Coh94] F.B.Cohen, *A short course on computer viruses* (2nd Ed.), J. Wiley, 1994.

[Cou] Crypto-Gram Newsletter, http://bt.counterpane.com/

[Cow03] C.Cowan, "Software security for open-source systems", *IEEE Security and Privacy*, January/February 2003, 38-45.

[csi] Computer Security Institute, www.gocsi.com

[dev] Devx, www.devx.com

[DoD] U.S. Department of Defense, *Trusted computer system evaluation criteria*, 1983.

[epi] Electronic Privacy Information Center (EPIC), http://www.epic.org

[Fer81] E. B. Fernandez, R. C. Summers, C. Wood, *Database Security and Integrity*, Addison-Wesley, Reading, Massachusetts, Systems Programming Series, February 1981, 320 pp., Japanese translation, 1982.

[Fer01] E.B.Fernandez, "An overview of Internet security", *Procs. of the World's Internet & Electronic Cities Conference (WIECC 2001)*, May 1-3, 2001, Kish Island, Iran.

[fir] FIRST, http://www.first.org

[Gol06] D. Gollmann, *Computer security (3nd Ed.)*, Wiley, 2011.

[goo] Google, http://directory.google.com/Top/Computers/Security/

[Gry05] D. Gryaznov, "Malware in popular networks", October 2005 http://www.mcafee.com/us/local_content/white_papers/threat_center/wp_dmitrygryaznov_vb2005.pdf

[How06] M. Howard and S. Lipner, *The Security Development Lifecycle*, Microsoft Press, 2006.

[How08] M. Howard, “Becoming a security expert”, *IEEE Security & Privacy*, Jan./Feb. 2008, 71-73.

[ics] ICSA Labs., http://www.icsa.net

[iee]   IEEE, *IEEE Transactions on Dependable and Secure Computing* (*TDSC),* http://www.computer.org/tdsc/

[its] IEEE Computer Society Tec. Comm. on Sec. and Privacy: www.ieee-security.org/

[iti] IT-ISAC, http://www.it-isac.org

[jpt] Journal of Privacy Technology, http://www.jopt.org

[Koo04] P. Koopman, “Embedded system security”, *Computer*, IEEE, July 2004, 95-07.

[Lan94] C. E. Landwehr , A. R. Bull, J.P. McDermott, and W.S.Choi, “A taxonomy of computer program security flaws, *ACM Comp. Surveys*, vol. 26, No 3, 1994, 211-254.

[Mar] P. Marques, *Analysis of software artifacts: Security*, set of slides, http://pmarques.dei.uc.pt/resources/SecurityIntro.pdf
A good description of a buffer overflow attack

[Mer03] R.T. Mercuri, “Standards insecurity”, *Comm. of the ACM*, vol. 46, No 12, December 2003, 21-25.

[Mel98] P.M. Melliar-Smith and L.E. Moser, “Surviving network partitioning”, *Computer*, IEEE, March 1998, 62-68.

[msd] MSDN, http://msdn.microsoft.com/

[neu] P. Neumann, http://www.csl.sri.com/users/neumann/neumann.html

[Neu95] P.G.Neumann, *Computer-related risks*, Addison-Wesley, 1995.

[Neu99] P.G. Neumann, “Risks of insiders”, *Comm. of the ACM*, vol. 42, No 12, December 1999, 160.

[nis] National Institute of Standards and Technology (NIST), *Common Criteria*, http://csrc.nist.gov/cc/

[OMG] OMG (Object Management Group): http://www.uml.org///www.omg.org/uml

[Pag07] F. Paget, “Identity theft”, McAfee White paper, January 2007. http://www.mcafee.com/us/local_content/white_papers/wp_id_theft_en.pdf

[Pea04] R. Pear and E. Lichtblau, “Administration sets forth a limited view on privacy”, *The New York Times*, March 6, 2004, A8.

[Per84] T.S.Perry and P. Wallich, “Can computer crime be stopped?”, *IEEE Spectrum*, May 1984, 34-45.

[Pfl11] C.P.Pfleeger, *Security in computing, 5th Ed*., Prentice-Hall, 2011.

[reg] The Register, http://www.theregister.co.uk/security

[san] SysAdmin, Audit, Network, Security Institute (SANS), www.sans.org
They organize conferences and publish a good newsletter about security. They also publish a yearly “Roadmap to Network Security’.

[Sch04] J. Schwartz and M. Maynard, “FBI got records on air travelers”, *The New York Times*, May 5, 2004, A1.

[Sch06] M. Schumacher, E. B.Fernandez, D. Hybertson, F. Buschmann, and P. Sommerlad, *Security Patterns: Integrating security and systems engineering"*, Wiley 2006.

[Sho82] J.F.Shoch and J.A.Hupp, “The “Worm” programs—Early experience with a distributed computation”, *Comm. of the ACM*, Vol. 25, No 3, March 1982, 172-180.

[Sko04] E. Skoudis, *Malware: Fighting malicious code*, Prentice Hall 2004.
This book has many details about malware, including history, ways of attack, available software, and references. It is somewhat verbose and makes no attempt to relate the cases to the conceptual structure of the computer.

[Sob01] D.L.Sobel, “Will Carnivore devour online privacy?”, Computer, IEEE May 2001, 87-88.

[Sta15] W. Stallings and L. Brown, *Computer security: Principles and practice (3rd Ed.)*, Pearson, 2015.

[Sta11] M. Stamp, *Information security* (2nd. Edition), J. Wiley 2011.
Heavy emphasis on cryptography, about 30 % of the book.

[Ste02] J. Steffan and M. Schumacher, “Collaborative attack modeling”, *Procs. of SAC 2002,* Madrid, Spain. http://www.ito.tu-darmstadt.de/publs/pdf/sac2002.pdf

[Sto09] B. Stone, “3 indicted in theft of 130 million card numbers”, *The New York Times*, August 18, 2009.

[Sum97] R.C.Summers, *Secure computing: Threats and safeguards*, McGraw-Hill, 1997.

[Sym] Symantec Antivirus Research Center : www.symantec.com/avcenter/index.html

[sys] Sys-con Media, http://www.sys-con.com/

[Tav04] H.T.Tavani, *Ethics and technology*, Wiley 2004.

[Uzu13] Anton V. Uzunov, Katrina Falkner, Eduardo B. Fernandez , "Distributed Software Architectures for the Determination and Incorporation of Security and Other Non-Functional Requirements", *22nd Australasian Software Engineering Conference (ASWEC 2013*), Melbourne, Australia, 4-7 June 2013

[Vau03] S.J. Vaughan-Nichols, "Saving private e-mail", *IEEE Spectrum*, August 2003, 40-44.

[Whit77] T. Whiteside, "Annals of crime—Dead souls in the computer", *The New Yorker*, Part I: August 22, 1977, 35-65 and Part II, August 29, 1977, 34-64.

[yah] Yahoo, http://dir.yahoo.com/Computers_and_Internet/Security_and_Encryption/