MODULE 4

Security has been a concern since the early days of computing, when a computer was isolated in a room and a threat could be posed only by malicious insiders. The Pandora's box[1] of threats opened wide once computers were able to communicate with one another. In an interconnected world, various embodiments of malware can migrate easily from one system to another, cross national borders, and infect systems all over the globe.

The security of computing and communication systems takes on a new urgency as society becomes increasingly dependent on the information infrastructure. Nowadays, even the critical infrastructure of a nation can be attacked by exploiting flaws in computer security. Malware, such as the Stuxnet virus, targets industrial control systems controlled by software [81]. Recently, the term *cyberwarfare* has entered the dictionary with the meaning "actions by a nation-state to penetrate another nation's computers or networks for the purposes of causing damage or disruption" [85].

A computer cloud is a target-rich environment for malicious individuals and criminal organizations. It is thus no surprise that security is a major concern for existing users and for potential new users of cloud computing services. In Section 3.10 we identified some of the security threats perceived by cloud users; in Section 9.1 we elaborate on this topic. Some of these risks are shared with other systems supporting network-centric computing and network-centric content, e.g., service-oriented architectures (SOAs), grids, and Web-based services.

Cloud computing is an entirely new approach to computing based on a new technology. It is therefore reasonable to expect that new methods to deal with some of the security threats will be developed, whereas other perceived threats will prove to be exaggerated. Indeed, "early on in the life cycle of a technology, there are many concerns about how this technology will be used . . . they represent a barrier to the acceptance . . . over the time, however, the concerns fade, especially if the value proposition is strong enough" [174].

The idea that moving to a cloud liberates an organization from many technical concerns related to computer security and eliminates internal threats is accepted by some members of the IT community. As we shall see throughout this chapter, this seems a rather naïve point of view, because outsourcing computing to a cloud generates major new security and privacy concerns. Moreover, service-level agreements do not provide adequate legal protection for cloud computer users, who are often left to deal with events beyond their control.

[1]In Greek mythology, Pandora was the first woman on Earth. When Prometheus stole fire from heaven, Zeus took vengeance by presenting Pandora to Epimetheus, the brother of Prometheus. Pandora was given a beautiful box and told not to open it under any circumstance. Impelled by curiosity, a trait given to her by the mischievous gods, Pandora opened the box, and all evil contained therein escaped, except for one item at the bottom of the box: Elpis, the Spirit of Hope.

One of the consequences of the breathtaking pace of development of information science and technology is that standards, regulations, and laws governing the activities of organizations supporting the new computing services, and in particular utility computing, have yet to be devised or adopted. As a result, many issues related to privacy, security, and trust in cloud computing are far from settled. The pool of resources of a cloud service provider can be dispersed over several countries or even several continents. Since information can freely cross national borders there is a need for international regulations to be adopted by the countries where data centers of cloud computing providers are located.

9.1 Cloud security risks

Some believe that it is very easy, possibly too easy, to start using cloud services without a proper understanding of the security risks and without the commitment to follow the ethics rules for cloud computing. A first question is: What are the security risks faced by cloud users? There is also the possibility that a cloud could be used to launch large-scale attacks against other components of the cyber infrastructure. The next question is: How can the nefarious use of cloud resources be prevented?

There are multiple ways to look at the security risks for cloud computing. A recent paper identifies three broad classes of risk [83]: traditional security threats, threats related to system availability, and threats related to third-party data control.

Traditional threats are those experienced for some time by any system connected to the Internet, but with some cloud-specific twists. The impact of traditional threats is amplified due to the vast amount of cloud resources and the large user population that can be affected. The fuzzy bounds of responsibility between the providers of cloud services and users and the difficulties in accurately identifying the cause of a problem add to cloud users' concerns.

The traditional threats begin at the user site. The user must protect the infrastructure used to connect to the cloud and to interact with the application running on the cloud. This task is more difficult because some components of this infrastructure are outside the firewall protecting the user.

The next threat is related to the authentication and authorization process. The procedures in place for one individual do not extend to an enterprise. In this case the cloud access of the members of an organization must be nuanced; individuals should be assigned distinct levels of privilege based on their roles in the organization. It is also nontrivial to merge or adapt the internal policies and security metrics of an organization with the ones of the cloud.

Moving from the user to the cloud, we see that the traditional types of attack have already affected cloud service providers. The favorite means of attack are distributed denial-of-service (DDoS) attacks, which prevent legitimate users accessing cloud services; phishing;[2] SQL injection;[3] or cross-site scripting.[4]

[2]Phishing is an attack aiming to gain information from a site database by masquerading as a trustworthy entity. Such information could be names and credit card numbers, Social Security Numbers (SSN), or other personal information stored by online merchants or other service providers.

[3]SQL injection is a form of attack typically used against a Web site. An SQL command entered in a Web form causes the contents of a database used by the Web site to be dumped to the attacker or altered. SQL injection can be used against other transaction-processing systems and is successful when the user input is not strongly typed and/or rigorously filtered.

[4]Cross-site scripting is the most popular form of attack against Web sites. A browser permits the attacker to insert client scripts into the Web pages and thus bypass the access controls at the Web site.

Cloud servers host multiple VMs, and multiple applications may run under each VM. Multitenency in conjunction with VMM vulnerabilities could open new attack channels for malicious users. Identifying the path followed by an attacker is much more difficult in a cloud environment. Traditional investigation methods based on digital forensics cannot be extended to a cloud, where the resources are shared among a large user population and the traces of events related to a security incident are wiped out due to the high rate of `write` operations on any storage media.

Availability of cloud services is another major concern. System failures, power outages, and other catastrophic events could shut down cloud services for extended periods of time. When such an event occurs, data lock-in, discussed in Section 3.5, could prevent a large organization whose business model depends on that data from functioning properly.

Clouds could also be affected by phase transition phenomena and other effects specific to complex systems (see Chapter 10). Another critical aspect of availability is that users cannot be assured that an application hosted on the cloud will return correct results.

Third-party control generates a spectrum of concerns caused by the lack of transparency and limited user control. For example, a cloud provider may subcontract some resources from a third party whose level of trust is questionable. There are examples when subcontractors failed to maintain the customer data. There are also examples when the third party was not a subcontractor but a hardware supplier and the loss of data was caused by poor-quality storage devices [83].

Storing proprietary data on a cloud is risky because cloud provider espionage poses real dangers. The terms of contractual obligations usually place all responsibilities for data security with the user. The Amazon Web Services customer agreement, for example, does not help boost user confidence as it states: "We . . . will not be liable to you for any direct, indirect, incidental . . . damages . . . nor . . . be responsible for any compensation, reimbursement, arising in connection with: (A) your inability to use the services . . . (B) the cost of procurement of substitute goods or services . . . or (D) any unauthorized access to, alteration of, or deletion, destruction, damage, loss or failure to store any of your content or other data."

It is very difficult for a cloud user to prove that data has been deleted by the service provider. The lack of transparency makes auditability a very difficult proposition for cloud computing. Auditing guidelines elaborated by the National Institute of Standards and Technology (NIST), such as the Federal Information Processing Standard (FIPS) and the Federal Information Security Management Act (FISMA), are mandatory for U.S. government agencies.

The first release of the Cloud Security Alliance (CSA) report in 2010 identifies seven top threats to cloud computing. These threats are the abuse of the cloud, APIs that are not fully secure, malicious insiders, shared technology, account hijacking, data loss or leakage, and unknown risk profiles [97]. According to this report, the *IaaS* delivery model can be affected by all threats. *PaaS* can be affected by all but the shared technology, whereas *SaaS* is affected by all but abuse and shared technology.

The term *abuse of the cloud* refers to the ability to conduct nefarious activities from the cloud – for example, using multiple *AWS* instances or applications supported by *IaaS* to launch DDoS attacks or to distribute spam and malware. *Shared technology* considers threats due to multitenant access supported by virtualization. VMMs can have flaws allowing a guest operating system to affect the security of the platform shared with other virtual machines.

Insecure APIs may not protect users during a range of activities, starting with authentication and access control to monitoring and control of the application during runtime. The cloud service providers do not disclose their hiring standards and policies; thus, the risks of *malicious insiders* cannot be ignored. The potential harm due to this particular form of attack is great.

Data loss or leakage are two risks with devastating consequences for an individual or an organization using cloud services. Maintaining copies of the data outside the cloud is often unfeasible due to the sheer volume of data. If the only copy of the data is stored on the cloud, sensitive data is permanently lost when cloud data replication fails and is followed by a storage media failure. Because some of the data often includes proprietary or sensitive data, access to such information by third parties could have severe consequences.

Account or service hijacking is a significant threat, and cloud users must be aware of and guard against all methods of stealing credentials. Finally, *unknown risk profile* refers to exposure to the ignorance or underestimation of the risks of cloud computing.

The 2011 version of the CSA report, "Security Guidance for Critical Area of Focus in Cloud Computing V3.0," provides a comprehensive analysis of and makes recommendations to minimize the risks inherent in cloud computing [98].

An attempt to identify and classify the attacks in a cloud computing environment is presented in [147]. The three actors involved in the model considered are the user, the service, and the cloud infrastructure, and there are six types of attacks possible (see Figure 9.1). The user can be attacked from two directions:

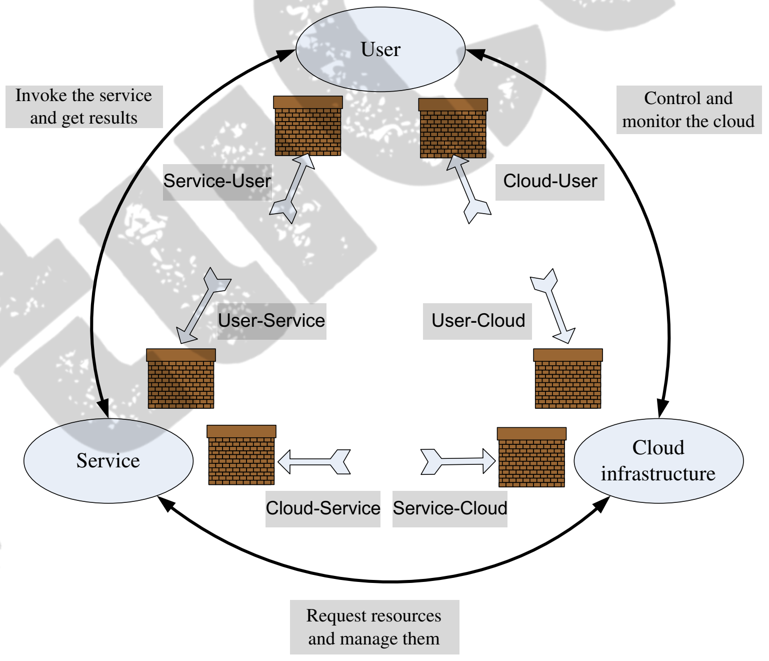

FIGURE 9.1

Surfaces of attacks in a cloud computing environment.

from the service and from the cloud. SSL certificate spoofing, attacks on browser caches, or phishing attacks are examples of attacks that originate at the service. The user can also be a victim of attacks that either originate at the cloud or spoofs that originate from the cloud infrastructure.

The service can be attacked from the user. Buffer overflow, SQL injection, and privilege escalation are the common types of attacks from the service. The service can also be subject to attack by the cloud infrastructure; this is probably the most serious line of attack. Limiting access to resources, privilege-related attacks, data distortion, and injecting additional operations are only a few of the many possible lines of attack originated at the cloud.

The cloud infrastructure can be attacked by a user who targets the cloud control system. The types of attack are the same ones that a user directs toward any other cloud service. The cloud infrastructure may also be targeted by a service requesting an excessive amount of resources and causing the exhaustion of the resources.

9.2 Security: The top concern for cloud users

Virtually all surveys report that security is the top concern for cloud users, who are accustomed to having full control of all systems on which sensitive information is stored and processed. Users typically operate inside a secure perimeter protected by a corporate firewall. In spite of the potential threats, users have to extend their trust to the cloud service provider if they want to benefit from the economical advantages of utility computing. This is a fairly difficult transition, yet it is a critical one for the future of cloud computing. To support this transition, some argue that cloud security is in the hands of experts, so users are even better protected than when they are in charge of their own security.

Major user concerns are unauthorized access to confidential information and data theft. Data is more vulnerable in storage than while it is being processed. Data is kept in storage for extended periods of time, whereas it is exposed to threats during processing for relatively short periods of time. Hence, close attention should be paid to the security of storage servers and to data in transit.

This does not mean that threats during processing can be ignored; such threats can originate from flaws in the VMM, rogue VMs, or a VMBR, as discussed in Section 5.12. There is also the risk of unauthorized access and data theft posed by rogue employees of a cloud service provider (CSP). The hiring and security screening policies of the CSP personnel are totally opaque processes to users, and this justifies users' concern about insider attacks.

The next concerns regard user control over the life cycle of data. It is virtually impossible for a user to determine whether data that should have been deleted is actually deleted. Even if it was deleted, there is no guarantee that the media was wiped and the next user is not able to recover confidential data. This problem is exacerbated because the CSPs rely on seamless backups to prevent accidental data loss. Such backups are done without users' consent or knowledge. During this exercise data records can be lost, accidentally deleted, or accessible to an attacker.

Lack of standardization is next on the list of concerns. Today there are no interoperability standards, as we discussed in Section 3.5. Many questions do not have satisfactory answers at this time. For example: What can be done when the service provided by the CSP is interrupted? How can we access our critically needed data in case of a blackout? What if the CSP drastically raises its prices? What is the cost of moving to a different CSP?

It is undeniable that auditing and compliance pose an entirely different set of challenges in cloud computing. These challenges are not yet resolved. A full audit trail on a cloud is an infeasible proposition at this time.

Another, less analyzed user concern is that cloud computing is based on a new technology expected to evolve in the future. Case in point: autonomic computing is likely to enter the scene. When this happens, self-organization, self-optimization, self-repair, and self-healing could generate additional security threats. In an autonomic system it will be even more difficult than at present to determine when an action occurred, what was the reason for that action, and how it created the opportunity for an attack or for data loss. It is still unclear how autonomic computing can be compliant with privacy and legal issues.

There is no doubt that multitenancy is the root cause of many user concerns. Nevertheless, multitenancy enables a higher server utilization thus, lower costs. Because it is one of the pillars of utility computing, users have to learn to live with multitenancy. The threats caused by multitenancy differ from one cloud delivery model to another. For example, in the case of *SaaS*, private information such as name, address, phone numbers, and possibly credit card numbers of many users is stored on one server, and when the security of that server is compromised, a large number of users are affected. We have already mentioned that multitenancy threats during processing time cannot be ignored.

Users are also greatly concerned about the legal framework for enforcing cloud computing security. The cloud technology has moved much faster than cloud security and privacy legislation, so users have legitimate concerns regarding the ability to defend their rights. Because the datacenters of a CSP may be located in several countries, it is difficult to understand which laws apply – the laws of the country where information is stored and processed, the laws of the countries where the information crossed from the user to the datacenter, or the laws of the country where the user is located.

To make matters even more complicated, a CSP may outsource the handling of personal and/or sensitive information. Existing laws stating that the CSP must exercise reasonable security may be difficult to implement in a case where there is a chain of outsourcing to companies in different countries. Finally, a CSP may be required by law to share private data with law enforcement agencies.

Now we examine briefly what cloud users can and should do to minimize security risks regarding data handling by the CSP. First, users should evaluate the security policies and the mechanisms the CSP has in place to enforce these policies. Then users should analyze the information that would be stored and processed on the cloud. Finally, the contractual obligations should be clearly spelled out.

The contract between the user and the CSP should do the following [290]:

1. State explicitly the CSP's obligations to securely handle sensitive information and its obligation to comply with privacy laws.
2. Spell out CSP liabilities for mishandling sensitive information.
3. Spell out CSP liabilities for data loss.
4. Spell out the rules governing the ownership of the data.
5. Specify the geographical regions where information and backups can be stored.

To minimize security risks, a user may try to avoid processing sensitive data on a cloud. The Secure Data Connector from Google carries out an analysis of the data structures involved and allows users to access data protected by a firewall. This solution is not feasible for several classes of application, e.g., processing of medical or personnel records. It may not be feasible when the cloud processing workflow requires cloud access to the entire volume of user data.

When the volume of sensitive data or the processing workflow requires sensitive data to be stored on a public or hybrid cloud, then, whenever feasible, data should be encrypted. This poses a dilemma because encryption prevents indexing and searching the data. For some applications it is possible to scramble the data to make it unintelligible to an intruder. Though this system is extremely inefficient, hence impractical at this time, it is possible to process encrypted data using either a fully homomorphic encryption scheme [134] or secure two-party computations [380].

9.3 Privacy and privacy impact assessment

The term *privacy* refers to the right of an individual, a group of individuals, or an organization to keep information of a personal or proprietary nature from being disclosed to others. Many nations view privacy as a basic human right. The Universal Declaration of Human Rights, Article 12, states: "No one shall be subjected to arbitrary interference with his privacy, family, home or correspondence, nor to attacks upon his honor and reputation. Everyone has the right to the protection of the law against such interference or attacks."

The U.S. Constitution contains no express right to privacy; however, the Bill of Rights reflects the concern of the framers for protecting specific aspects of privacy.[5] In the United Kingdom privacy is guaranteed by the Data Protection Act. The European Court of Human Rights has developed many documents defining the right to privacy.

At the same time, the right to privacy is limited by laws. For example, taxation laws require individuals to share information about personal income or earnings. Individual privacy may conflict with other basic human rights, e.g., freedom of speech. Privacy laws differ from country to country; laws in one country may require public disclosure of information considered private in other countries and cultures.

The digital age has confronted legislators with significant challenges related to privacy as new threats have emerged. For example, personal information voluntarily shared, but stolen from sites granted access to it or misused, can lead to *identity theft*.

Some countries have been more aggressive than others in addressing the new privacy concerns. For example, the countries of the European Union (EU) have very strict laws governing handling of personal data in the digital age. A sweeping new privacy right, the "right to be forgotten," is codified as part of a broad new proposed data protection regulation in the EU. This right addresses the following problem: Today it is very hard to escape your past when every photo, status update, and tweet lives forever on some Web site.

Our discussion targets primarily public clouds where privacy has an entirely new dimension because the data, often in an unencrypted form, resides on servers owned by a CSP. Services based on individual preferences, the location of individuals, membership in social networks, or other personal information

[5]The First Amendment covers the protection of beliefs, the Third Amendment privacy of homes, the Fourth Amendment the privacy of person and possessions against unreasonable searches, the Fifth Amendment the privilege against self-incrimination and thus, the privacy of personal information. According to some Justices, the Ninth Amendment, which reads, "The enumeration in the Constitution, of certain rights, shall not be construed to deny or disparage others retained by the people," can be viewed as a protection of privacy in ways not explicitly specified by the first eight amendments in the Bill of Rights.

present a special risk. The owner of the data cannot rely exclusively on the CSP to guarantee the privacy of the data.

Privacy concerns are different for the three cloud delivery models and also depend on the actual context. For example, consider Gmail, a widely used *SaaS* delivery model. Gmail privacy policy reads (see `www.google.com/policies/privacy/`, accessed on October 6, 2012): "We collect information in two ways: information you give us ... like your name, email address, telephone number or credit card; information we get from your use of our services such as: ... device information, ... log information, ... location information, ... unique application numbers, ... local storage, ... cookies and anonymous identifiers ... We will share personal information with companies, organizations or individuals outside of Google if we have a good-faith belief that access, use, preservation or disclosure of the information is reasonably necessary to: meet any applicable law, regulation, legal process or enforceable governmental request ... protect against harm to the rights, property or safety of Google, our users or the public as required or permitted by law. We may share aggregated, nonpersonally identifiable information publicly and with our partners like publishers, advertisers or connected sites. For example, we may share information publicly to show trends about the general use of our services."

The main aspects of privacy are: the lack of user control, potential unauthorized secondary use, data proliferation, and dynamic provisioning [290]. The lack of user control refers to the fact that user-centric data control is incompatible with cloud usage. Once data is stored on the CSP's servers, the user loses control of the exact location, and in some instances the user could lose access to the data. For example, in case of the Gmail service, the account owner has no control over where the data is stored or how long old emails are stored in some backups of the servers.

A CSP may obtain revenues from unauthorized secondary usage of the information, e.g., for targeted advertising. There are no technological means to prevent this use. Dynamic provisioning refers to threats due to outsourcing. A range of issues is very fuzzy; for example, how to identify the subcontractors of a CSP, what rights to the data they have, and what rights to data are transferable in case of bankruptcy or merger.

There is a need for legislation addressing the multiple aspects of privacy in the digital age. A document elaborated by the Federal Trade Commission for the U.S. Congress states [122]: "Consumer-oriented commercial Web sites that collect personal identifying information from or about consumers online would be required to comply with the four widely accepted fair information practices:

1. *Notice.* Web sites would be required to provide consumers clear and conspicuous notice of their information practices, including what information they collect, how they collect it (e.g., directly or through nonobvious means such as cookies), how they use it, how they provide Choice, Access, and Security to consumers, whether they disclose the information collected to other entities, and whether other entities are collecting information through the site.
2. *Choice.* Web sites would be required to offer consumers choices as to how their personal identifying information is used beyond the use for which the information was provided (e.g., to consummate a transaction). Such choice would encompass both internal secondary uses (such as marketing back to consumers) and external secondary uses (such as disclosing data to other entities).
3. *Access.* Web sites would be required to offer consumers reasonable access to the information a Web site has collected about them, including a reasonable opportunity to review information and to correct inaccuracies or delete information.

4. *Security.* Web sites would be required to take reasonable steps to protect the security of the information they collect from consumers. The Commission recognizes that the implementation of these practices may vary with the nature of the information collected and the uses to which it is put, as well as with technological developments. For this reason, the Commission recommends that any legislation be phrased in general terms and be technologically neutral. Thus, the definitions of fair information practices set forth in the statute should be broad enough to provide flexibility to the implementing agency in promulgating its rules or regulations."

There is a need for tools capable of identifing privacy issues in information systems, the so-called *Privacy Impact Assesment (PIA)*. As of mid-2012 there were no international standards for such a process, though different countries and organizations require PIA reports. An example of an analysis is to assess the legal implications of the U.K.-U.S. Safe Harbor process to allow U.S. companies to comply with the European Directive 95/46/EC[6] on the protection of personal data.

Such an assessment forces a proactive attitude toward privacy. An ab-initio approach to embedding privacy rules in new systems is preferable to painful changes that could affect the functionality of existing systems.

A PIA tool that could be deployed as a Web-based service is proposed in [345]. The inputs to the tool includes project information, an outline of project documents, privacy risks, and stakeholders. The tool will produce a PIA report consisting of a summary of findings, a risk summary, security, transparency, and cross-border data flows.

The centerpiece of the PIA tool is a knowledge base (KB) created and maintained by domain experts. The users of the *SaaS* service providing access to the PIA tool must fill in a questionnaire. The system uses templates to generate additional questions necessary and to fill in the PIA report. An expert system infers which rules are satisfied by the facts in the database and provided by the users and executes the rule with the highest priority.

9.4 Trust

Trust in the context of cloud computing is intimately related to the general problem of trust in online activities. In this section we first discuss the traditional concept of trust and then the trust necessary to online activities.

According to the Merriam-Webster dictionary, *trust* means "assured reliance on the character, ability, strength, or truth of someone or something." Trust is a complex phenomenon; it enables cooperative behavior, promotes adaptive organizational forms, reduces harmful conflict, decreases transaction costs, facilitates formulation of ad hoc workgroups, and promotes effective responses to crisis [309].

Two conditions must exist for trust to develop. The first condition is *risk*, the perceived probability of loss; indeed, trust would not be necessary if there were no risk involved, if there is a certainty that an action can succeed. The second condition is *interdependence*, the idea that the interests of one entity cannot be achieved without reliance on other entities. A trust relationship goes though three phases: (1) a building phase, when trust is formed; (2) a stability phase, when trust exists; and (3) a dissolution phase, when trust declines.

[6]See eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31995L0046:en:HTML.

There are different reasons for and forms of trust. Utilitarian reasons could be based on the belief that the costly penalties for breach of trust exceed any potential benefits from opportunistic behavior. This is the essence of *deterrence-based* trust. Another reason is the belief that the action involving the other party is in the self-interest of that party. This is the so-called *calculus-based* trust. After a long sequence of interactions, *relational trust* between entities can develop based on the accumulated experience of dependability and reliance on each other.

The common wisdom is that an entity must work very hard to build trust but may lose that trust very easily; a single violation of trust can lead to irreparable damage. *Persistent trust* is trust based on the long-term behavior of an entity, whereas *dynamic trust* is based on a specific context, e.g., a state of the system or the effect of technological developments.

The trust in the Internet "obscures or lacks entirely the dimensions of character and personality, nature of relationship, and institutional character" of traditional trust [258]. The missing identity, personal characteristics, and role definitions are elements we have to deal with in the context of online trust.

The Internet offers individuals the ability to obscure or conceal their identities. The resulting anonymity reduces the cues normally used in judgments of trust. The identity is critical for developing trust relations; it allows us to base our trust on the past history of interactions with an entity. Anonymity causes mistrust because identity is associated with accountability and, in the absence of identity, accountability cannot be enforced. The opacity extends immediately from identity to personal characteristics. It is impossible to infer whether the entity or individual we transact with is who it pretends to be, since the transactions occur between entities separated in time and distance. Finally, there are no guarantees that the entities we transact with fully understand the role they have assumed.

To remedy the loss of clues, we need security mechanisms for access control, transparency of identity, and surveillance. The mechanisms for access control are designed to keep intruders and mischievous agents out. Identity transparency requires that the relationship between a virtual agent and a physical person should be carefully checked through methods such as biometric identification. Digital signatures and digital certificates are used for identification. Surveillance could be based on *intrusion detection* or on logging and auditing. The first option is based on real-time monitoring, the second on offline sifting through audit records.

Credentials are used when an entity is not known. Credentials are issued by a trusted authority and describe the qualities of the entity using the credential. A Doctor of Dental Surgery diploma hanging on the wall of a dentist's office is a credential that the individual has been trained by an accredited university and hence is capable of performing a set of dental procedures; similarly, a digital signature is a credential used in many distributed applications.

Policies and *reputation* are two ways of determining trust. Policies reveal the conditions to obtain trust and the actions to take when some of the conditions are met. Policies require the verification of credentials. Reputation is a quality attributed to an entity based on a relatively long history of interactions with or possibly observations of the entity. Recommendations are based on trust decisions made by others and filtered through the perspective of the entity assessing the trust.

In a computer science context, "trust of a party A to a party B for a service X is the measurable belief of A in that B behaves dependably for a specified period within a specified context (in relation to service X)" [272]. An assurance about the operation of a particular hardware or software component leads to persistent social-based trust in that component. A comprehensive discussion of trust in computer services in the semantic Web can be found in [26]. In Section 11.10 we discuss the concept of trust in the context of cognitive radio networks where multiple transmitters compete for free communication channels. In Section 11.11 we present a cloud-based trust management service.

9.5 Operating system security

An operating system (OS) allows multiple applications to share the hardware resources of a physical system, subject to a set of policies. A critical function of an OS is to protect applications against a wide range of malicious attacks such as unauthorized access to privileged information, tempering with executable code, and spoofing. Such attacks can now target even single-user systems such as personal computers, tablets, or smartphones. Data brought into the system may contain malicious code; this could occur via a Java applet, or data imported by a browser from a malicious Web site.

The *mandatory security* of an OS is considered to be "any security policy where the definition of the policy logic and the assignment of security attributes is tightly controlled by a system security policy administrator" [209]. Access control, authentication usage, and cryptographic usage policies are all elements of mandatory OS security. The first policy specifies how the OS controls the access to different system objects, the second defines the authentication mechanisms the OS uses to authenticate a principal, and the last specifies the cryptographic mechanisms used to protect the data. A necessary but not sufficient condition for security is that the subsystems tasked with performing security-related functions are temper-proof and cannot be bypassed. The OS should confine an application to a unique security domain.

Applications with special privileges that perform security-related functions are called *trusted applications.* Such applications should only be allowed the lowest level of privileges required to perform their functions. For example, type enforcement is a mandatory security mechanism that can be used to restrict a trusted application to the lowest level of privileges.

Enforcing mandatory security through mechanisms left to the discretion of users could lead to a breach of security due not only to malicious intent but also carelessness or lack of understanding. Discretionary mechanisms place the burden of security on individual users. Moreover, an application may change a carefully defined discretionary policy without the consent of the user, whereas a mandatory policy can only be changed by a system administrator.

Unfortunately, commercial operating systems do not support multilayered security; such systems only distinguish between a completely privileged security domain and a completely unprivileged one. Some operating systems, such as *Windows NT*, allow a program to inherit all the privileges of the program invoking it, regardless of the level of trust in that program.

The existence of *trusted paths*, mechanisms supporting user interactions with trusted software, is critical to system security. If such mechanisms do not exist, malicious software can impersonate trusted software. Some systems provide trust paths for a few functions such as login authentication and password changing and allow servers to authenticate their clients.

The solution discussed in [209] is to decompose a complex mechanism into several components with well-defined roles. For example, the access control mechanism for the application space could consist of *enforcer* and *decider* components. To access a protected object, the enforcer will gather the required information about the agent attempting the access and will pass this information to the decider, together with the information about the object and the elements of the policy decision. Finally, it will carry out the actions requested by the decider.

A trusted-path mechanism is required to prevent malicious software invoked by an authorized application to tamper with the attributes of the object and/or with the policy rules. A trusted path is also required to prevent an impostor from impersonating the decider agent. A similar solution is proposed for cryptography usage, which should be decomposed into an analysis of the invocation mechanisms and an analysis of the cryptographic mechanism.

Another question is how an OS can protect itself and the applications running under it from malicious mobile code attempting to gain access to the data and the other resources and compromise system confidentiality and/or integrity. Java Security Manager uses the type-safety attributes of Java to prevent unauthorized actions of an application running in a “sandbox.” Yet, the Java Virtual Machine (JVM) accepts byte code in violation of language semantics; moreover, it cannot protect itself from tampering by other applications.

Even if all these security problems could be eliminated, good security relies on the ability of the file system to preserve the integrity of Java class code. The approach to require digitally signed applets and accept them only from trusted sources could fail due to the all-or-nothing security model. A solution to securing mobile communication could be to confine a browser to a distinct security domain.

Specialized *closed-box platforms* such as the ones on some cellular phones, game consoles, and automated teller machines (ATMs) could have embedded cryptographic keys that allow themselves to reveal their true identity to remote systems and authenticate the software running on them. Such facilities are not available to *open-box platforms*, the traditional hardware designed for commodity operating systems.

A highly secure operating system is necessary but not sufficient unto itself; application-specific security is also necessary. Sometimes security implemented above the operating system is better. This is the case for electronic commerce that requires a digital signature on each transaction.

We conclude that commodity operating systems offer low assurance. Indeed, an OS is a complex software system consisting of millions of lines of code, and it is vulnerable to a wide range of malicious attacks. An OS poorly isolates one application from another, and once an application is compromised, the entire physical platform and all applications running on it can be affected. The platform security level is thus reduced to the security level of the most vulnerable application running on the platform.

Operating systems provide only weak mechanisms for applications to authenticate to one another and do not have a trusted path between users and applications. These shortcomings add to the challenges of providing security in a distributed computing environment. For example, a financial application cannot determine whether a request comes from an authorized user or from a malicious program; in turn, a human user cannot distinguish a response from a malicious program impersonating the service from the response provided by the service itself.

9.6 Virtual machine security

The hybrid and the hosted VM models in Figures 5.3(c) and (d), respectively, expose the entire system to the vulnerability of the host operating system; thus, we will not analyze these models. Our discussion of virtual machine security is restricted to the traditional system VM model in Figure 5.3(b), where the VMM controls access to the hardware.

Virtual security services are typically provided by the VMM, as shown in Figure 9.2(a). Another alternative is to have a dedicated security services VM, as shown in Figure 9.2(b). A secure *trusted*

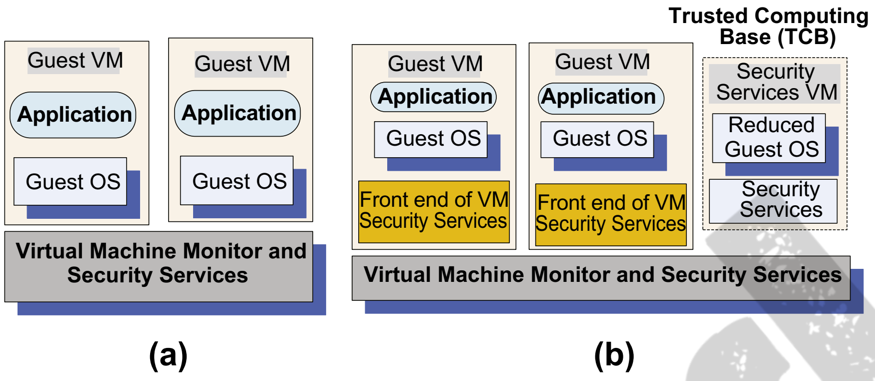
Trusted Computing Base (TCB)
Guest VM
Application
Guest OS
Guest VM
Application
Guest OS
Virtual Machine Monitor and Security Services
Guest VM
Application
Guest OS
Front end of VM Security Services
Guest VM
Application
Guest OS
Front end of VM Security Services
Security Services VM
Reduced Guest OS
Security Services
Virtual Machine Monitor and Security Services
(a)
(b)

FIGURE 9.2

(a) Virtual security services provided by the VMM. (b) A dedicated security VM.

computing base (TCB) is a necessary condition for security in a virtual machine environment; if the TCB is compromised, the security of the entire system is affected.

The analysis of *Xen* and *vBlades* in Sections 5.8 and 5.10 shows that VM technology provides a stricter isolation of virtual machines from one another than the isolation of processes in a traditional operating system. Indeed, a VMM controls the execution of privileged operations and can thus enforce memory isolation as well as disk and network access. The VMMs are considerably less complex and better structured than traditional operating systems; thus, they are in a better position to respond to security attacks. A major challenge is that a VMM sees only raw data regarding the state of a guest operating system, whereas security services typically operate at a higher logical level, e.g., at the level of a file rather than a disk block.

A guest OS runs on simulated hardware, and the VMM has access to the state of all virtual machines operating on the same hardware. The state of a guest virtual machine can be saved, restored, cloned, and encrypted by the VMM. Not only can replication ensure reliability, it can also support security, whereas cloning could be used to recognize a malicious application by testing it on a cloned system and observing whether it behaves normally. We can also clone a running system and examine the effect of potentially dangerous applications. Another interesting possibility is to have the guest VM's files moved to a dedicated VM and thus, protect it from attacks [389]; this is possible because inter-VM communication is faster than communication between two physical machines.

Sophisticated attackers are able to fingerprint virtual machines and avoid VM *honeypots* designed to study the methods of attack. They can also attempt to access VM-logging files and thus recover sensitive data; such files have to be very carefully protected to prevent unauthorized access to cryptographic keys and other sensitive data.

There is no free lunch; thus, we expect to pay some price for the better security provided by virtualization. This price includes: higher hardware costs, because a virtual system requires more resources, such as CPU cycles, memory, disk, and network bandwidth; the cost of developing VMMs and modifying the host operating systems in case of paravirtualization; and the overhead of virtualization because the VMM is involved in privileged operations.

A recent paper [389] surveys VM-based intrusion detection systems such as `Livewire` and `Siren`, which exploit the three capabilities of a virtual machine for intrusion detection: isolation, inspection, and interposition. We have examined isolation; inspection means that the VMM has the ability to review the state of the guest VMs, and interposition means that the VMM can trap and emulate the privileged instruction issued by the guest VMs. The paper also discusses VM-based intrusion prevention systems such as `SVFS, NetTop`, and `IntroVirt` and surveys `Terra`, a VM-based trust computing platform. `Terra` uses a *trusted virtual machine monitor* to partition resources among virtual machines.

The security group involved with the NIST project has identified the following VMM- and VM-based threats:

- VMM-based threats:
  1. Starvation of resources and denial of service for some VMs. Probable causes: (a) badly configured resource limits for some VMs; (b) a rogue VM with the capability to bypass resource limits set in the VMM.
  2. VM side-channel attacks. Malicious attacks on one or more VMs by a rogue VM under the same VMM. Probable causes: (a) lack of proper isolation of inter-VM traffic due to misconfiguration of the virtual network residing in the VMM; (b) limitation of packet inspection devices to handle high-speed traffic, e.g., video traffic; (c) presence of VM instances built from insecure VM images, e.g., a VM image having a guest OS without the latest patches.
  3. Buffer overflow attacks.
- VM-based threats:
  1. Deployment of rogue or insecure VM. Unauthorized users may create insecure instances from images or may perform unauthorized administrative actions on existing VMs. Probable cause: improper configuration of access controls on VM administrative tasks such as instance creation, launching, suspension, reactivation, and so on.
  2. Presence of insecure and tampered VM images in the VM image repository. Probable causes: (a) lack of access control to the VM image repository; (b) lack of mechanisms to verify the integrity of the images, e.g., digitally signed image.

9.8 Security risks posed by shared images

Even when we assume that a cloud service provider is trustworthy, many users either ignore or underestimate the danger posed by other sources of concern. One of them, especially critical to the *IaaS* cloud delivery model, is image sharing. For example, a user of *AWS* has the option to choose between Amazon Machine Images (AMIs), accessible through the *Quick Start* or the *Community AMI* menus of the *EC2* service. The option of using one of these AMIs is especially tempting for a first-time or less sophisticated user.

First, let's review the process to create an AMI. We can start from a running system, from another AMI, or from the image of a VM and copy the contents of the file system to the *S3*, the so-called *bundling*. The first of the three steps in bundling is to create an image, the second step is to compress and encrypt the image, and the last step is to split the image into several segments and then upload the segments to the *S3*.

Two procedures for the creation of an image are available: `ec2-bundle-image` and `ec2-bundle-volume`. The first is used for images prepared as loopback files[10] when the data is transferred to the image in blocks. To bundle a running system, the creator of the image can use the second procedure when bundling works at the level of the file system and files are copied recursively to the image.

To use an image, a user has to specify the resources, provide the credentials for login, provide a firewall configuration, and specify the region, as discussed in Section 3.1. Once instantiated, the user is informed about the public DNS and the virtual machine is made available. A *Linux* system can be accessed using `ssh` at port 22, whereas the Remote Desktop at port 3389 is used for *Windows*.

A recent paper reports on the results of an analysis carried out over a period of several months, from November 2010 to May 2011, of over 5,000 AMIs available through the public catalog at Amazon [38].

[10] A *loopback file system* (LOFS) is a virtual file system that provides an alternate path to an existing file system. When other file systems are mounted onto an LOFS file system, the original file system does not change. One useful purpose of LOFS is to take a CD-ROM image file, a file of type iso, and mount it on the file system and then access it without the need to record a CD-R. It is somewhat equivalent to the *Linux* `mount -o loop` option but adds a level of abstraction; most commands that apply to a device can be used to handle the mapped file.

Many of the analyzed images allowed a user to *undelete* files and recover credentials, private keys, or other types of sensitive information with little effort and using standard tools. The results of this study were shared with Amazon's Security Team, which acted promptly to reduce the threats posed to *AWS* users.

The details of the testing methodology can be found in [38]. Here we only discuss the results. The study was able to audit some 5,303 images out of the 8,448 *Linux* AMIs and 1,202 *Windows* AMIs at Amazon sites in the United States, Europe, and Asia. The audit covered software vulnerabilities and security and privacy risks.

The average duration of an audit was 77 minutes for a *Windows* image and 21 minutes for a *Linux* image; the average disk space used was about 1 GB and 2.7 GB, respectively. The entire file system of a *Windows* AMI was audited because most malware targets *Windows* systems. Only directories containing executables for *Linux* AMIs were scanned; this strategy and the considerably longer start-up time of *Windows* explain the time discrepancy of the audits across the types of AMIs.

The *software vulnerability* audit revealed that 98% of the *Windows* AMIs (249 out of 253) and 58% of *Linux* AMIs (2,005 out of 3,432) audited had critical vulnerabilities. The average number of vulnerabilities per AMI were 46 for *Windows* and 11 for *Linux*. Some of the images were rather old; 145, 38, and 2 *Windows* AMIs and 1,197, 364, and 106 *Linux* were older than two, three, and four years, respectively. The tool used to detect vulnerabilities, *Nessus*, available from `www.tenable.com/productus/nessus`, classifies the vulnerabilities based on their severity in four groups, at levels 0–3. The audit reported only vulnerabilities of the highest severity level, e.g., remote code execution.

Three types of *security risks* were analyzed: (1) backdoors and leftover credentials, (2) unsolicited connections, and (3) malware. An astounding finding is that about 22% of the scanned *Linux* AMIs contained credentials allowing an intruder to remotely log into the system. Some 100 passwords, 995 `ssh` keys, and 90 cases in which both passwords and keys could be retrieved were identified.

To rent a *Linux* AMI, a user must provide the public part of the `ssh` key, and this key is stored in the `authorized_keys` in the home directory. This opens a backdoor for a malicious creator of an AMI who does not remove his own public key from the image and can remotely log into any instance of this AMI. Another backdoor is opened when the `ssh` server allows password-based authentication and the malicious creator of an AMI does not remove his own password. This backdoor is opened even wider as one can extract the password hashes and then crack the passwords using a tool such as John the Ripper (see `www.openwall.com/john`).

Another threat is posed by the omission of the `cloud-init` script that should be invoked when the image is booted. This script, provided by Amazon, regenerates the host key an `ssh` server uses to identify itself; the public part of this key is used to authenticate the server. When this key is shared among several systems, these systems become vulnerable to *man-in-the middle*[11] attacks. When this

[11]In a *man-in-the-middle* an attacker impersonates the agents at both ends of a communication channel and makes them believe that they communicate through a secure channel. For example, if B sends her public key to A, but C is able to intercept it, such an attack proceeds as follows: C sends a forged message to A claiming to be from B but instead includes C's public key. Then A encrypts his message with C's key, believing that he is using B's key, and sends the encrypted message to B. The intruder, C, intercepts, deciphers the message using her private key, possibly alters the message, and re-encrypts the public key B originally sent to A. When B receives the newly encrypted message, she believes it came from A.

script does not run, an attacker can use the *NMap* tool[12] to match the `ssh` keys discovered in the AMI images with the keys obtained via *NMap*. The study reports that the authors were able to identify more than 2,100 instances following this procedure.

Unsolicited connections pose a serious threat to a system. Outgoing connections allow an outside entity to receive privileged information, e.g., the IP address of an instance and events recorded by a *syslog* daemon to files in the *var/log* directory of a *Linux* system. Such information is available only to users with administrative privileges. The audit detected two *Linux* instances with modified *syslog* daemons, which forwarded to an outside agent information about events such as login and incoming requests to a Web server. Some of the unsolicited connections are legitimate – for example, connections to a software update site. It is next to impossible to distinguish legitimate from malicious connections.

Malware, including viruses, worms, spyware, and trojans, were identified using *ClamAV*, a software tool with a database of some 850,000 malware signatures, available from `www.clamav.net`. Two infected *Windows* AMIs were discovered, one with a *Trojan-Spy* (variant 50112) and a second one with a *Trojan-Agent* (variant 173287). The first trojan carries out keylogging and allows stealing data from the files system and monitoring processes; the AMI also included a tool called *Trojan.Firepass* to decrypt and recover passwords stored by the *Firefox* browser.

The creator of a shared AMI assumes some *privacy risks*; his private keys, IP addresses, browser history, shell history, and deleted files can be recovered from the published images. A malicious agent can recover the *AWS* API keys that are not password protected. Then the malicious agent can start AMIs and run cloud applications at no cost to herself, since the computing charges are passed on to the owner of the API key. The search can target files with names such as $pk - [0 - 9A - Z]^*.pem$ or $cert - [0 - 9A - Z]^*.pem$ used to store API keys.

Another avenue for a malicious agent is to recover `ssh` keys stored in files named *id_dsa* and *id_rsa*. Though `ssh` keys can be protected by a *passphrase*,[13] the audit determined that the majority of *ssh* keys (54 out of 56) were not password protected.

Recovery of IP addresses of other systems owned by the same user requires access to the *lastlog* or the *lastb* databases. The audit found 187 AMIs with a total of more than 66,000 entries in their *lastb* databases. Nine AMIs contained Firefox browser history and allowed the auditor to identify the domains contacted by the user.

In addition, 612 AMIs contained at least one shell history file. The audit analyzed 869 history files named *~/.history*, *~/.bash_history*, and *~/.sh_history*, containing some, 160,000 lines of command history, and identified 74 identification credentials. Users should be aware that when *HTTP* is used to transfer information from a user to a Web site, the *GET* requests are stored in the logs of the Web server. Passwords and credit card numbers communicated via a *GET* request can be exploited by a malicious agent with access to such logs. When remote credentials such as the DNS management password are available, a malicious agent can redirect traffic from its original destination to her own system.

[12]*NMap* is a security tool running on most operating systems, including *Linux, Microsoft Windows, Solaris, HP-UX, SGI-IRIX*, and BSD variants such as *Mac OS X*, to map the network. *Mapping the network* means discovering hosts and services in a network.

[13]A *passphrase* is a sequence of words used to control access to a computer system; it is the analog of a password but provides added security. For high-security nonmilitary applications, NIST recommends an 80-bit-strength passphrase. Hence a secure passphrase should consist of at least 58 characters, including uppercase and alphanumeric characters. The entropy of written English is less than 1.1 bits per character.

Recovery of deleted files containing sensitive information poses another risk for the provider of an image. When the sectors on the disk containing sensitive information are actually overwritten by another file, recovery of sensitive information is much harder. To be safe, the creator of the image effort should use utilities such as `shred`, `scrub`, `zerofree`, or `wipe` to make recovery of sensitive information next to impossible. If the image is created with the block-level tool discussed at the beginning of this section, the image will contain blocks of the file system marked as free; such blocks may contain information from deleted files. The audit process was able to recover files from 98% of the AMIs using the `exundelete` utility. The number of files recovered from an AMI was as low as 6 and as high as 40,000.

We conclude that the users of published AMIs as well as the providers of images may be vulnerable to a wide range of security risks and must be fully aware of the dangers posed by image sharing.

9.9 Security risks posed by a management OS

We often hear that virtualization enhances security because a virtual machine monitor or hypervisor is considerably smaller than an operating system. For example, the *Xen* VMM discussed in Section 5.8 has approximately 60,000 lines of code, one to two orders of magnitude fewer than a traditional operating system.[14]

A hypervisor supports stronger isolation between the VMs running under it than the isolation between processes supported by a traditional operating system. Yet the hypervisor must rely on a management OS to create VMs and to transfer data in and out from a guest VM to storage devices and network interfaces.

A small VMM can be carefully analyzed; thus, one could conclude that the security risks in a virtual environment are diminished. We have to be cautious with such sweeping statements. Indeed, the trusted computer base (TCB)[15] of a cloud computing environment includes not only the hypervisor but also the management OS. The management OS supports administrative tools, live migration, device drivers, and device emulators.

For example, the TCB of an environment based on *Xen* includes not only the hardware and the hypervisor but also the management operating system running in the so-called *Dom0* (see Figure 9.3). System vulnerabilities can be introduced by both software components, *Xen*, and the management operating system. An analysis of *Xen* vulnerabilities reports that 21 of the 23 attacks were against service components of the control VM [90]; 11 attacks were attributed to problems in the guest OS caused by buffer overflow[16] and 8 were denial-of-service attacks.

Dom0 manages the building of all user domains (*DomU*), a process consisting of several steps:

1. Allocate memory in the *Dom0* address space and load the kernel of the guest operating system from secondary storage.
2. Allocate memory for the new VM and use foreign mapping[17] to load the kernel to the new VM.

[14]The number of lines of code of the *Linux* operating system evolved in time from 176,250 for *Linux 1.0.0*, released in March 1995, to 1,800,847 for *Linux 2.2.0*, released in January 1999; 3,377,902 for *Linux 2.4.0*, released in January 2001; and to 5,929,913 for *Linux 2.6.0*, released in December 2003.
[15]The TCB is defined as the totality of protection mechanisms within a computer system, including hardware, firmware, and software, the combination of which is responsible for enforcing a security policy.
[16]Buffer overflow allows execution of arbitrary code in a privileged mode.
[17]The foreign mapping mechanism of *Xen* is used by *Dom0* to map arbitrary memory frames of a VM into its page tables.

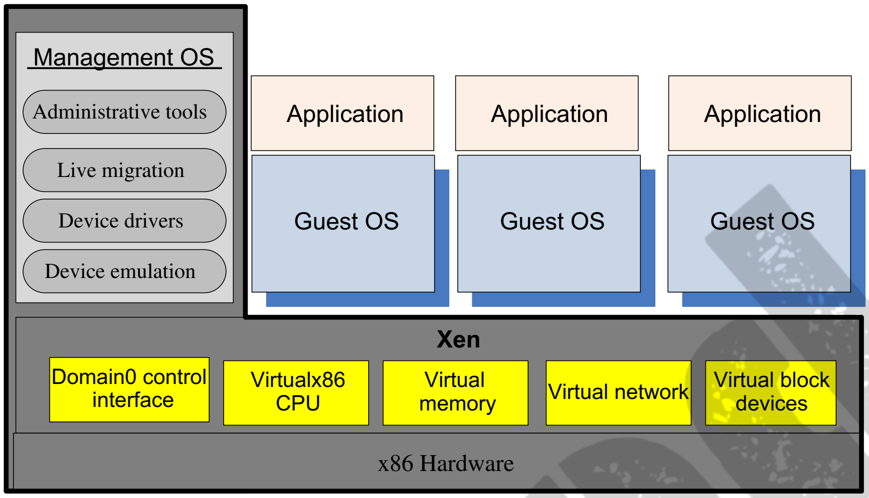

FIGURE 9.3

The trusted computing base of a *Xen*-based environment includes the hardware, *Xen*, and the management operating system running in *Dom0*. The management OS supports administrative tools, live migration, device drivers, and device emulators. A guest operating system and applications running under it reside in a *DomU*.

3. Set up the initial page tables for the new VM.
4. Release the foreign mapping on the new VM memory, set up the virtual CPU registers, and launch the new VM.

A malicious *Dom0* can play several nasty tricks at the time when it creates a *DomU* [215]:

- Refuse to carry out the steps necessary to start the new VM, an action that can be considered a *denial-of-service* attack.
- Modify the kernel of the guest operating system in ways that will allow a third party to monitor and control the execution of applications running under the new VM.
- Undermine the integrity of the new VM by setting the wrong page tables and/or setting up incorrect virtual CPU registers.
- Refuse to release the foreign mapping and access the memory while the new VM is running.

Let us now turn our attention to the run-time interaction between *Dom0* and a *DomU*. Recall that *Dom0* exposes a set of abstract devices to the guest operating systems using *split drivers*. The front end of such a driver is in the *DomU* and its back end in *Dom0*, and the two communicate via a ring in shared memory (see Section 5.8).

In the original implementation of *Xen* a service running in a *DomU* sends data to or receives data from a client located outside the cloud using a network interface in *Dom0*; it transfers the data to I/O devices using a device driver in *Dom0*.[18] Therefore, we have to ensure that run-time communication

[18]Later implementations of *Xen* offer the pass-through option.

through *Dom0* is encrypted. Yet, *Transport Layer Security* (TLS) does not guarantee that *Dom0* cannot extract cryptographic keys from the memory of the OS and applications running in *DomU*.

A significant security weakness of *Dom0* is that the entire state of the system is maintained by *XenStore* (see Section 5.8). A malicious VM can deny access to this critical element of the system to other VMs; it can also gain access to the memory of a *DomU*. This brings us to additional requirements for confidentiality and integrity imposed on *Dom0*.

Dom0 should be prohibited from using foreign mapping for sharing memory with a *DomU* unless a *DomU* initiates the procedure in response to a hypercall from *Dom0*. When this happens, *Dom0* should be provided with an encrypted copy of the memory pages and of the virtual CPU registers. The entire process should be closely monitored by the hypervisor, which, after the access, should check the integrity of the affected *DomU*.

A virtualization architecture that guarantees confidentiality, integrity, and availability for the TCB of a *Xen*-based system is presented in [215]. A secure environment when *Dom0* cannot be trusted can only be ensured if the guest application is able to store, communicate, and process data safely. Thus, the guest software should have access to secure secondary storage on a remote storage server for keeping sensitive data and network interfaces to communicate with the user. We also need a secure run-time system.

To implement a secure run-time system we have to intercept and control the hypercalls used for communication between a *Dom0* that cannot be trusted and a *DomU* we want to protect. Hypercalls issued by *Dom0* that do not `read` or `write` to the memory of a *DomU* or to its virtual registers should be allowed. Other hypercalls should be restricted either completely or during specific time *windows*. For example, hypercalls used by *Dom0* for debugging or for the control of the IOMMU[19] should be prohibited.

We cannot restrict some of the hypercalls issued by *Dom0*, even though they can be harmful to the security of a *DomU*. For example, foreign mapping and access to the virtual registers are needed to save and restore the state of a *DomU*. We should check the integrity of a *DomU* after the execution of such security-critical hypercalls.

New hypercalls are necessary to protect:

- The privacy and integrity of the virtual CPU of a VM. When *Dom0* wants to save the state of the VM, the hypercall should be intercepted and the contents of the virtual CPU registers should be encrypted. When a *DomU* is restored, the virtual CPU context should be decrypted and then an integrity check should be carried out.
- The privacy and integrity of the VM virtual memory. The *page table update* hypercall should be intercepted and the page should be encrypted so that *Dom0* handles only encrypted pages of the VM. To guarantee integrity, the hypervisor should calculate a hash of all the memory pages before they are saved by *Dom0*. Because a restored *DomU* may be allocated a different memory region, an address translation is necessary (see [215]).
- The freshness of the virtual CPU and the memory of the VM. The solution is to add to the hash a version number.

As expected, the increased level of security and privacy leads to increased overhead. Measurements reported in [215] show increases by factors of 1.7 to 2.3 for the domain build time, 1.3 to 1.5 for the domain save time, and 1.7 to 1.9 for the domain restore time.

[19]An input/output memory management unit (IOMMU) connects main memory with a DMA-capable I/O bus. It maps device-visible virtual addresses to physical memory addresses and provides memory protection from misbehaving devices.