CHAPTER

CLOUD SECURITY 11

Security has been a concern since the early days of computing when a computer was isolated, and threats could only be posed by someone with access to the computer room. Once computers were able to communicate with one another the Pandora box of threats was wide opened. 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 world.

Security of computer and communication systems takes on a new urgency as the society becomes increasingly more dependent on the information infrastructure. Even the critical infrastructure of a nation can be attacked by exploiting flaws in computer security and often human naivety. Malware, such as the Stuxnet virus, targets industrial systems controlled by software [104]. The concept of *cyberwarfare* meaning "actions by a nation-state to penetrate another nation's computers or networks for the purposes of causing damage or disruption" [111], was recently included in the dictionary.

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. Some of the security risks faced by computer clouds are shared with other systems supporting network-centric computing and network-centric content, e.g., service-oriented architectures, 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, while other perceived threats will prove to be exaggerated [30]. 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" [245].

The breathtaking pace of developments in information science and technology has many side-effects. One of them 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 adopted. As a result, many issues related to privacy, security, and trust in cloud computing are far from being settled. For example, there are no international regulations related to data security and privacy. Data stored on a computer cloud can freely cross national borders among the data centers of the CSP.

The chapter starts with a discussion of cloud users concerns related to security in Section 11.1 then, in Section 11.2 we elaborate on the security threats perceived by cloud users already mentioned in Section 2.11. Privacy and trust are covered in Sections 11.3 and 11.4. Encryption protects data in cloud storage, but data must be decrypted for processing as discussed in Section 11.5. Threats during processing originating from flaws in the hypervisors, rogue VMs, or a VMBR, discussed in Section 10.13, cannot be ignored.

The analysis of database service security in Section 11.6 is followed by a presentation of operating system security, VM security, and security of virtualization in Sections 11.7, 11.8, and 11.9, respectively. Sections 11.10 and 11.11 analyze the security risks posed by shared images and by a

Cloud Computing. DOI: 10.1016/B978-0-12-812810-7.00015-7

management OS. An overview of the Xoar hypervisor, a version of Xen which breaks the monolithic Design of the TCB, is discussed in Section 11.12 followed in Section 11.13 by a presentation of a trusted hypervisor and mobile device security in Section 11.14.

11.1 SECURITY, THE TOP CONCERN FOR CLOUD USERS

Some believe that moving to a computer cloud frees an organization from all concerns related to computer security and eliminates a wide range of threats to data integrity. They believe that cloud security is in the hands of experts, hence cloud users are better protected than when using their own computing resources. As we shall see throughout this chapter, these views are not entirely justified.

Outsourcing computing to a cloud generates major new security and privacy concerns. Moreover, the Service Level Agreements do not provide adequate legal protection for cloud computer users who are often left to deal with events beyond their control.

Some cloud users were accustomed to operate inside a secure perimeter protected by a corporate firewall. Now they have to extend their trust to the cloud service provider if they wish to benefit from the economical advantages of utility computing. The transition from a model when users have full control of all systems where their sensitive information is stored and processed is a difficult one. The reality is that virtually all surveys report that security is the top concern of cloud users.

Major user concerns are about the unauthorized access to confidential information and the data theft. Data is more vulnerable in storage, than while it is being processed. Data is kept in storage for extended periods of time, while during processing it is exposed to threats for relatively short time. Close attention should be paid to storage server security and to data in transit. There is also the risk of unauthorized access and data theft posed by rogue employees of a CSP. Cloud users are concerned about insider attacks because hiring and security screening policies of a CSP are totally opaque to the outsiders.

The next concerns regard the user control over the lifecycle of data. It is virtually impossible for a user to determine if data that should have been deleted was actually deleted. Even if deleted, there is no guarantee that the media was wiped out and the next user is not able to recover confidential data. This problem is exacerbated as the CSPs rely on seamless backups to prevent accidental data loss. Such backups are done without user knowledge or consent. 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 inter-operability standards as discussed in Section 2.7. Important questions do not have satisfactory answers, e.g.: What can be done when service provided by the CSP is interrupted? How to access 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 unfeasible 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 the present time 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 multi-tenancy is the root cause of many user concerns. Nevertheless, multi-tenancy enables a higher server utilization, thus lower costs. The users have to learn to live with multi-tenancy, one of the pillars of utility computing. The threats caused by multi-tenancy differ from one cloud delivery model to another. For example, in case of SaaS private information such as name, address, phone numbers, possibly credit card numbers of many users are stored on one server; when the security of that server is compromised a large number of users are affected.

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 thus, users have legitimate concerns regarding the ability to defend their rights. The data centers of a CSP may be located in several countries and it is unclear what laws apply, the laws of the country where information is stored and processed, the laws of the countries the information crossed when sent by the user, or the laws of the user's country.

To make matter even more complicated, a CSP may outsource handling of personal and/or sensitive information. Existing laws stating that the CSP must exercise reasonable security may be difficult to implement in case when there is a chain of outsourcing to companies in different countries. Lastly, a CSP may be required by law to share private data with law enforcement agencies. For example, Microsoft was served a subpoena to provide emails exchanges by users of the Hotmail service.

The question is: What cloud users can and should do to minimize the security risks regarding the data handling by the CSP? First, a user should evaluate the security policies and the mechanisms the CSP has in place to enforce these policies. Then, the user 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 a user and a CSP should [400] state clearly:

1. CSPs obligations to handle sensitive information and its obligation to comply with privacy laws.
2. CSP liabilities for mishandling sensitive information, e.g., data loss.
3. The rules governing ownership of the data.
4. Specify the geographical regions where information and backups can be stored.

To minimize the security risks a user may try to avoid processing sensitive data on a cloud. The Secure Data Connector from Google caries out an analysis of the data structures involved and allows access to data protected by a firewall. This solution is not feasible for several classes of applications, e.g., processing of medical or personnel records. The solution 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 the cloud then, whenever feasible, data should be encrypted [189] and [534].

11.2 CLOUD SECURITY RISKS

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

The next question is: What are the security risks faced by cloud users? There are multiple ways to look at the cloud security risks. A recent paper identifies three broad classes [109]: 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 long list of cloud user concerns includes also the fuzzy bounds of responsibility between the providers of cloud services and users, as well as the difficulties to accurately identify the cause of a problem.

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 authentication and authorization. Procedures in place for one individual do not extend to an enterprise, the cloud access of the members of an organization must be nuanced. Different individuals should be assigned distinct levels of privilege based on their role 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.

Traditional attacks have already affected cloud service providers. The favorite means of attack are: distributed denial of service (DDDS) attacks which prevent legitimate users to access cloud services, phishing, SQL injection, or cross-site scripting. Phishing aims to gain information from a database by masquerading as a trustworthy entity. Such information could be names and credit card numbers, social security numbers, other personal information stored by online merchants or by other service providers.

SQL injection is 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 either dumped to the attacker or altered. SQL injection can be used against other transaction processing systems and it is successful when the user input is not strongly typed or rigorously filtered. 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.

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

Availability of cloud services is another major concern. System failures, power outages, and other catastrophic events could shutdown cloud services for extended periods of time. Data lock-in discussed in Section 2.7 could prevent a large organization whose business model depends on these data to function properly, when such a rare event occurs.

Clouds can also be affected by phase transition phenomena and other effects specific to complex systems. Another critical aspect of availability is that the users cannot be assured that an application hosted on the cloud returns correct results.

Third-party control generates a spectrum of concerns caused by 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 [109].

Storing proprietary data on the cloud is risky as 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 does not help user's 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 (NIST) such as the Federal Information Processing Standard (FIPS) and the Federal Information Security Management Act (FISMA) are mandatory for US Government agencies.

The 2010 Cloud Security Alliance (CSA) report. The report identifies seven top threats to cloud computing. These threats are: the abusive use of the cloud, APIs that are not fully secure, malicious insiders, shared technology, account hijacking, data loss or leakage, and unknown risk profile [123]. According to this report the IaaS delivery model can be affected by all threats. PaaS can be the affected by all, but the shared technology, while SaaS is affected by all, but abuse and shared technology.

Abusing the cloud refers to conducting nefarious activities from the cloud. For example, use multiple AWS instances or applications supported by IaaS to launch distributed denial of service attacks or to distribute spam and malware. *Shared technology* considers threats due to multi-tenant access supported by virtualization. Hypervisors can have flaws allowing a guest OS to affect the security of the platform shared with other VMs.

Insecure APIs may not protect the 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 attacks is high.

Data loss and data 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, then sensitive data is permanently lost when cloud data replication fails followed by a storage media failure. As 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 to steal credentials. Lastly, *unknown risk profile* refers to exposure to the ignorance or underestimation of the risks of cloud computing.

The 2011 CSA report. The report "Security Guidance for Critical Area of Focus in Cloud Computing V3.0," provides a comprehensive analysis of the risks and makes recommendations to minimize the risk in cloud computing [124].

An attempt to identify and classify the attacks in a cloud computing environment is discussed in [207]. 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 11.1. The user can be attacked

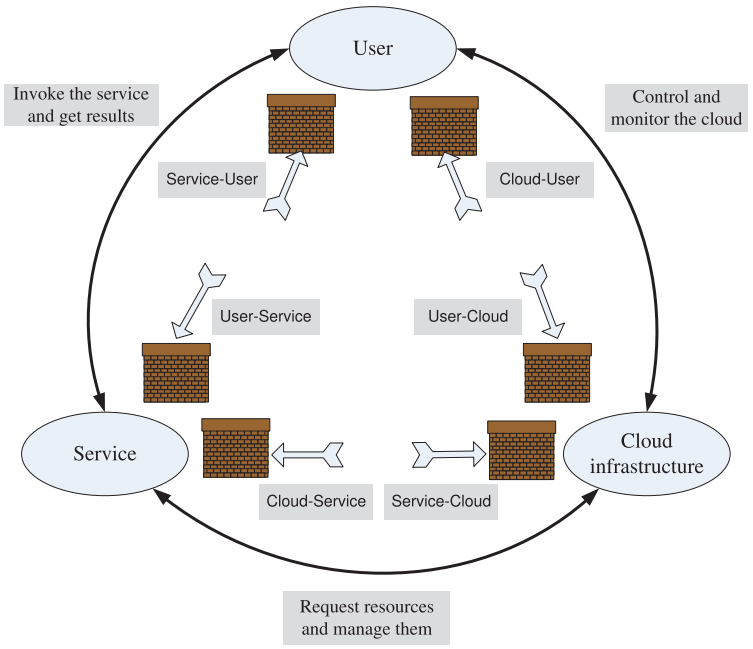

FIGURE 11.1

Surfaces of attacks in a cloud computing environment.

from two directions, the service and the cloud. Secure Sockets Layer (SSL) certificate spoofing, attacks on browser caches, or phishing attacks are example of attacks that originate at the service. The user can also be a victim of attacks that either truly originate or that spoof originating from the cloud infrastructure.

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

The cloud infrastructure can be attacked by a user which targets the cloud control system. These types of attacks are the same a user would direct 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.

Top twelve cloud security threats. The 2016 CSA report lists the top security threats [414]:

1. Data breaches. The most damaging breaches are for sensitive data including financial and health information, trade secrets, and intellectual property. The ultimate responsibility rests with the organizations maintaining data on the cloud and CSA recommends that organizations use multi-factor authentication and encryption to protect against data breaches. Multi-factor authentication such as one-time passwords, phone-based authentication, and smart card protection make it harder for attackers to use stolen credentials.

2. Compromised credentials and broken authentication. Such attacks are due to lax authentication, weak passwords, and poor key and/or certificate management.
3. Hacked interfaces and APIs. Cloud security and service availability can be compromised by a weak API. When third parties rely on APIs more services and credentials are exposed.
4. Exploited system vulnerabilities. Resource sharing and multi-tenancy create new attack surfaces but the cost to discover and repair vulnerabilities is small compared to the potential damage.
5. Account hijacking. All accounts should be monitored so that every transaction can be traced to the individual requesting it.
6. Malicious insiders. This threat can be difficult to detect and system administrator errors could sometimes be falsely diagnosed as threats. A good policy is to segregate duties and enforce activities such as logging, monitoring, and auditing administrator activities.

The other six threats are: advanced persistent threats (APTs), permanent data loss, inadequate diligence, cloud service abuse, DoS attacks, and shared technology.

An update of the "Cloud Controls Matrix" spells out the impact of control specifications on cloud architecture, cloud delivery models, and other aspects of the cloud ecosystem, according to https://cloudsecurityalliance.org/download/cloud-controls-matrix-v3-0-1/.

Cloud vulnerability incidents reported over a period of four years and data breach incidents in 2014 identified several other threats including: hardware failures, natural disasters, cloud-related malware, inadequate infrastructure design and planning, point-of-sale (POS) intrusions and payment card skimmers, crimeware and cyber-espionage, insider and privilege misuse, web app attacks, and physical theft/loss [480].

11.3 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 personal nature or proprietary information from being disclosed. 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.[1] 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, the taxation laws require individuals to share information about personal income or earnings. Individual privacy may conflict with other basic human rights e.g., with freedom of speech. The privacy laws differ from country to

[1]The 1st Amendment covers the protection of beliefs, the 3rd Amendment privacy of homes, the 4th Amendment the privacy of person and possessions against unreasonable searches, the 5th Amendment the privilege against self-incrimination, thus the privacy of personal information, and, according to some Justices, the 9th Amendment that 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.

country; laws in one country may require public disclosure of information considered private in other countries and cultures.

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 in addressing the new privacy concerns than others. For example, European Union (EU) has 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 EU. This right addresses the problem that it is hard to escape your past now 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 as data, often in an un-encrypted form, resides on servers owned by a CSP. Services based on individual preferences, location of individuals, membership in social networks, or other personal information 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 the widely used Gmail; Gmail privacy policy reads (see http://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, non-personally 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 cloud privacy are: the lack of user control, potential unauthorized secondary use, data proliferation, and dynamic provisioning [400]. 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 servers of the CSP the user losses control on the exact location, and in some instances it could lose access to the data. For example, in case of the Gmail service the account owner has no control on where the data is stored or how long old Emails are stored on 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 are 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 the need for legislation addressing the multiple aspects of privacy in the digital age. A document elaborated by the Federal Trading Commission for the US Congress states [172]: "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 should 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 non-obvious 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 should 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 choices 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 the need for tools capable to identify privacy issues in information systems, the so called *Privacy Impact Assessment (PIA)*. As of mid 2017 there are no international standards for such a process, though different countries and organization require PIA reports. An example of an analysis is to assess the legal implications of the UK-US Safe Harbor process to allow US companies to comply with the European Directive 95/46/EC[2] on the protection of personal data.

Such an assessment forces a proactive attitude towards privacy. An ab initio approach for 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 web-based service is proposed in [478]. The input 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-borders data flows.

The centerpiece of a 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 to fill in the PIA report. An expert system infers which rules are satisfied by the facts in the database as provided by the users and executes the rule with the highest priority.

11.4 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 in online activities.

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

Trust. 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 work groups, and promotes effective responses to crisis [430].

Two conditions must exist for trust to develop. The first is *risk*, the perceived probability of loss. Indeed, trust would not be necessary if there is no risk involved, if there is a certainty that an action can succeed. The second is *interdependence*, the interests of one entity cannot be archived without reliance on other entities.

A trust relationship goes though three phases:

1. Building phase, when trust is formed.
2. Stability phase, when trust exists.
3. Dissolution phase, when trust declines.

There are different reasons 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 developed 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 trust very easily. A single violation of trust can lead to irreparable damages. *Persistent trust* is based on the long term behavior of an entity, while *dynamic trust* is based on a specific context, e.g., state of the system or the effect of technological developments.

Internet trust. Internet trust "obscures or lacks entirely the dimensions of character and personality, nature of relationship, and institutional character" of the traditional trust [360]. 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 identity. The resulting anonymity reduces the clues normally used in judgments of trust. 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 absence of identity accountability cannot be enforced.

The opacity extends immediately from identity to personal characteristics. It is impossible to infer if the entity or individual we transact with is who it pretends to be, as the transactions occur between entities separated in time and distance. Lastly, 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 relation 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 can be based on logging and auditing. The first is based on real-time monitoring, while the second relies on off-line sifting through audit records.

Credentials are used when the entity is not known; credentials are issued by a trusted authority and describe the qualities of the entity using the credential. A DDS (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 capable to perform a set of procedures. 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 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 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 that B behaves dependably for a specified period within a specified context (in relation to service X)," [376]. An assurance about the operation of 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 [38]. In Section B.1 we discuss the concept of trust in the context of cognitive radio networks where multiple transmitters compete for communication channels. Then, in Section B.3 we present a cloud-based trust management service.

11.5 CLOUD DATA ENCRYPTION

The government, large corporations, and individual users ponder if it safe to store sensitive information on a public cloud. Encryption is the obvious solution to protect outsourced data and cloud service providers have been compelled to offer encryption services. For example, Amazon offers AWS Key Management Service (KMS) to create and control the encryption keys used by clients to encrypt their data. KMS is integrated with other AWS services including EBS, S3, RDS, Redshift, Elastic Transcoder, and WorkMail. AWS also offers Encryption SDK for developers.

The seminal RSA paper [424] and the survey of existing public-key crypto systems in [433] are some of the notable publications in the vast literature dedicated to cryptosystems. Several new research results in cryptography are important to data security in cloud computing. In 1999 Pascal Paillier proposed a trapdoor mechanism based on composite residuosity classes, i.e., factoring a hard-to-factor number $n = pq$ where p and q are two large prime numbers [388]. This solution exploits the homomorphic properties of composite residuosity classes to design distributed cryptographic protocols. A major breakthrough are the algorithms for Fully Homomorphic Encryption (FHE) proposed by Craig Gentry in his seminal 2009 dissertation at Stanford University [189,190]. In recent years, searchable symmetric encryption protocols have been reported in [86] and [168].

Homomorphic encryption. Sensitive data is safe while in storage, provided that it is encrypted with strong encryption. But encrypted data must be decrypted for processing and this opens a window of vulnerability. So a first question examined in this section is if it is feasible to operate on encrypted data. The *homomorphic encryption*, a long time dream of security experts, reflects the concept of homomorphism, a structure-preserving map $f(\cdot)$ between two algebraic structures of the same type see Figure 11.2.

When $f(\cdot)$ is a one-to-one mapping, call $f^{-1} : A' \rightarrow A$ the inverse of $f(\cdot)$. Then $a = f^{-1}(a')$, $b = f^{-1}(b')$, $c = f^{-1}(c')$. In this case we can carry out the composition operation $\diamond$ in the target domain

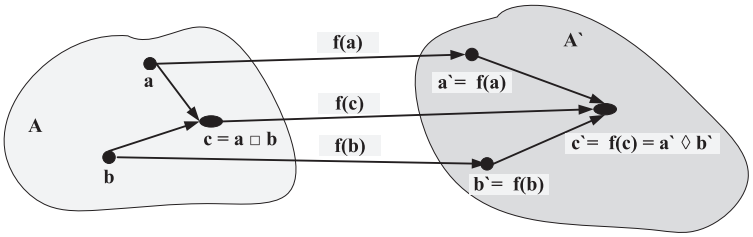

FIGURE 11.2

A homomorphism $f : A \rightarrow A'$ is a structure-preserving map between sets A and A' with the composition operations $\square$ and $\diamond$, respectively. Let $a, b, c \in A$ with $c = a \square b$ and $a', b', c' \in A'$ with $c' = a' \diamond b'$. Let $a' = f(a), b' = f(b), c' = f(c)$ be the results of the mapping $f(\cdot)$. If f is a homomorphism, then the composition operation $\diamond$ in the target domain A' produces the same result as mapping the result of the operation $\square$ applied to the two elements in the original domain A: $f(a) \diamond f(b) = f(a \square b)$.

and apply the inverse mapping to get the same result produced by the $\square$ composition operation in the original domain, $f^{-1}(a) \diamond f^{-1}(b) = f(a \square b)$, as shown in Figure 11.2.

In case of homomorphic encryption the mapping $f(\cdot)$ is a one-to-one transformation, the encryption procedure; its inverse, $f^{-1}(\cdot)$ is the decryption procedure and the composition operation can be any arithmetic and logic operation carried out with encrypted data. In this case we can carry arithmetic and/or logic operations with encrypted data and the decryption of the result of these operations is identical with the result of carrying out the same operations with the plaintext data. The window of vulnerability created when data is decrypted for processing disappears.

General computations with encrypted data are theoretically feasible using FHE algorithms. Unfortunately, the homomorphic encryption is not a practical solution at this time. Existing algorithms for homomorphic encryption increase the processing time with encrypted data by many orders of magnitude compared with processing of plaintext data. A recent implementation of FHE [218] requires about six minutes per batch; the processing time for a simple operation on encrypted data dropped to almost one second after improvements in other experiments [154].

Users send a variety of queries to many large databases stored on clouds. Such queries often involve logic and arithmetic functions so an important question is if it is feasible and practical to search encrypted databases. Application of widely used encryption techniques to database systems could lead to significant performance degradation. For example, if an entire column of a NoSQL database table contains sensitive information and it is encrypted, then a query predicate with a comparison operator requires a scan of the entire table to evaluate the query. This is due to the fact that existing encryption algorithms do not preserve order and database indices such as B-tree can no longer be used.

Order Preserving Encryption. OPE can be used for encryption of numeric data, it maps a range of numerical values into a much larger and sparse range of values [70]. Let a order-preserving function $f : \{1, ..., M\} \rightarrow \{1, ..., N\}$ with $N >> M$ be uniquely represented by a combination of M out of N ordered items. Given N balls in a bin, M black and $N - M$ white, we draw a ball at random without replacement at each step. The random variable X describing the total number of balls in our sample after we collect the k-th black ball follows the negative hypergeometric distribution (NHG). One can

show that a order preserving $f(x)$ for a given point $x \in \{1, \ldots, M\}$ has a NHG distribution over a random choice of f.

To encrypt plaintext x the OPE encryption algorithm performs a binary search down to x. Given the secret key K the algorithm first assigns $Encrypt(K, M/2)$, then $Encrypt(K, M/4)$ if the index $m < M/2$ and $Encrypt(K, 3M/4)$ otherwise, and so on, until $Encrypt(K, x)$ is assigned. Each ciphertext assignment is made according to the output of the negative hypergeometric sampling algorithm. One can prove by strong induction on the size of the plaintext space that the resulting scheme induces a random order-preserving function from the plaintext to ciphertext space.

To allow efficient range queries on encrypted data, it is sufficient to have an order-preserving hash function family H (not necessarily invertible). The OPE algorithm would use a secret key $(K_{Encrypt}, K_H)$ where $K_{Encrypt}$ is a key for a normal (randomized) encryption scheme and K_H is a key for H. Then $Encrypt(K_{Encrypt}, x) \,||\, H(K_H, x)$ will be the encryption of x [70].

Searching encrypted databases is of particular interest [11]. Several types of searches are frequently conducted including: single-keyword, multi-keyword, fuzzy-keyword, ranked, authorized, and verifiable search. Searchable symmetric encryption (SSE) is used when an encrypted databases $\mathcal{E}$ is outsourced to a cloud or to a different organization. SSE hides information about the database and the queries.

The client only stores the cryptographic key. To search the database the client encrypts the query, sends it to the database server, receives the encrypted result of the query and decrypts it using the cryptographic key. The information leakage from these searches is confined to query patterns, while disclosure of explicit data and query plaintext values is prevented.

An SSE protocol supporting conjunctive search and general Boolean queries on symmetrically encrypted data was proposed in [86]. This SSE protocol scales to very large databases. It can be used for arbitrarily structured data including free text search with the moderate and well defined leakage to the outsourced server. Performance results of a prototype applied to encrypted search over the entire English Wikipedia are reported. The protocol was extended with support for range, substring, wildcard, and phrase queries [168].

The next question is if sensitive data stored on the servers of a private cloud is vulnerable. The threat posed by an outsider attacker is diminished if the private cloud is protected by an effective firewall. Nevertheless, there are dangers posed by an insider. If such an attacker has access to log files it can infer the location of database hot spots, copy data selectively, and use the data for a nefarious activity. To minimize the risks posed by an insider, a set of protections rings should be enforced to restrict the access of each member of the staff to a limited area of the data base.

11.6 SECURITY OF DATABASE SERVICES

Cloud users often delegate control of their data to the database services supported by virtually all CSP and are concerned with security aspects of DBaaS. The model used to evaluate DBaaS security includes several groups of entities: data owners, users of data, CSPs, and third party agents or Third Party Auditors (TPAs).

Data owners and DBaaS users fear compromised integrity and confidentiality, as well as data unavailability. Insufficient authorization, authentication and accounting mechanisms, inconsistent use of

encryption keys and techniques, alteration or deletion of records without maintaining backup, and operational failures are the major causes of data loss in DBaaS.

Some data integrity and privacy issues are due to the absence of authentication, authorization and accounting controls, or poor key management for encryption and decryption. Confidentiality means that only authorized users should have access to the data. Unencrypted data is vulnerable to bugs, errors, and attacks from external entities affecting data confidentiality. Insider attacks are another concern for DBaaS users and data owners. Superusers have unlimited privileges and misuse of superuser privileges poses a considerable threat to confidential data such as medical records, sensitive business data, proprietary product data, and so on.

Malicious external attackers use spoofing, sniffing, man-in-the-middle attacks, side channeling and illegal transactions to launch DoS attacks. Another concern is illegal recovery of data from storage devices, a side effect of multi-tenancy. CSP often carry out sanitation operations after deleting data from physical devices, but sophisticated attackers can still recover information from storage devices, unless a thorough scrubbing operation is carried out. Data is also vulnerable during transfer from the data owner to the DBaaS through public networks. Encryption before data transmission can reduce the risks posed to the data in transit to the cloud.

Data provenance, the process of establishing the origin of data and its movement between databases, uses metadata to determine the data accuracy, but the security assessments are time-sensitive. Moreover, analyzing large provenance metadata graphs is computationally expensive.

Cloud users are not aware of the physical location of their data. This lack of transparency allows cloud service providers to optimize the use of resources but in case of security breaches it is next to impossible for users to identify compromised resources. DBaaS users do not have fine-grained control of the remote execution environment and cannot inspect the execution traces to detect the occurrence of illegal operations.

To increase availability, performance and to enhance reliability, cloud database services replicate data. Ensuring consistency among the replicas is challenging. Another critical function of DBaaS is to carry out timely backups of all sensitive and confidential data to facilitate quick recovery in case of disasters. Auditing and monitoring are important functions of a DBaaS but generate their own security risks when delegated to TPAs. Conventional methods for auditing and monitoring demand detailed knowledge of the network infrastructure and physical devices. Data privacy laws can be violated as consumers are unaware where the data is actually stored. Privacy laws in Europe and South America prohibit storing data outside the country of origin.

In summary, DBaaS data *availability* is affected by several threats including:

- Resource exhaustion caused by imprecise specification of user needs or incorrect evaluation of user specifications.
- Failures of the consistency management; multiple hardware and/or software failures lead to inconsistent views of user data.
- Failure of the monitoring and auditing system.

DBaaS data *confidentiality* is affected by insider and outsider attacks, access control issues, illegal data recovery from storage, network breaches, third-party access, inability to establish the provenance of the data.

11.7 OPERATING SYSTEM SECURITY

An operating system 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, tampering with executable code, and spoofing. Such attacks can target even single-user systems such as personal computers, tablets, or smart phones. Data brought in the system may contain malicious code; this could be the case of a Java applet, or of data imported by a browser from a malicious web site.

The *mandatory security* of an OS is considered to be [295]: "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." Access control, authentication usage, and cryptographic usage policies are all elements of the mandatory OS security.

Access control policies specify how OS controls access to different system objects, authentication usage defines the authentication mechanisms used by the OS to authenticate a principal, and cryptographic usage policies specify the cryptographic mechanisms used to protect the data. A necessary but not sufficient condition for security is that the subsystems tasked to perform security-related functions are tamper-proof and cannot be bypassed. An OS should confine an application to a unique security domain.

Applications with special privileges performing 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 at user's discretion can lead to a breach of security, sometimes due to malicious intent, in other cases due to carelessness, or to 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, while a mandatory policy can only be changed by a system administrator.

Unfortunately, commercial operating systems do not support multi-layered security. They only distinguish between a completely privileged security domain and a completely unprivileged one. Some operating systems, e.g., 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 for system security. When such mechanisms do not exist malicious software can impersonate trusted software. Some systems allow servers to authenticate their clients and provide trusted paths for a few functions, such as login authentication and password changing.

A solution to the trusted path problem is to decompose a complex mechanism in several components with well-defined roles [295]. 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, 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 the 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 applications running under it from malicious mobile code attempting to gain access to data and 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 from other applications.

Even if these security problems could be eliminated, the security relies on the ability of the file system to preserve the integrity of the Java class code. Requiring digitally signed applets and accepting them only from trusted sources could fail due to the all-or-nothing security model. A solution to securing mobile communications 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 Automatic 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. Application-specific security is also necessary. Sometimes, security implemented above the operating system is better, e.g., electronic commerce 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; 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 one another and do not have a trusted path between users and applications. These shortcomings add to the challenges of providing security in a distribute computing environment. For example, a financial application cannot determine if 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.

11.8 VIRTUAL MACHINE SECURITY

The following discussion of VM security is restricted to the traditional system VM model in Figure 10.1B when a hypervisor controls the access to the hardware. The hybrid and the hosted VM models shown in Figures 10.1C and D, respectively, expose the entire system to the vulnerability of the host operating system thus, will not be analyzed.

Virtual security services are typically provided by the hypervisor as shown in Figure 11.3A; another alternative is to have a dedicated VM providing security service as in Figure 11.3B. A secure TCB (Trusted Computing Base) is a necessary condition for security in a VM environment. When the TCB is compromised then the security of the entire system is affected.

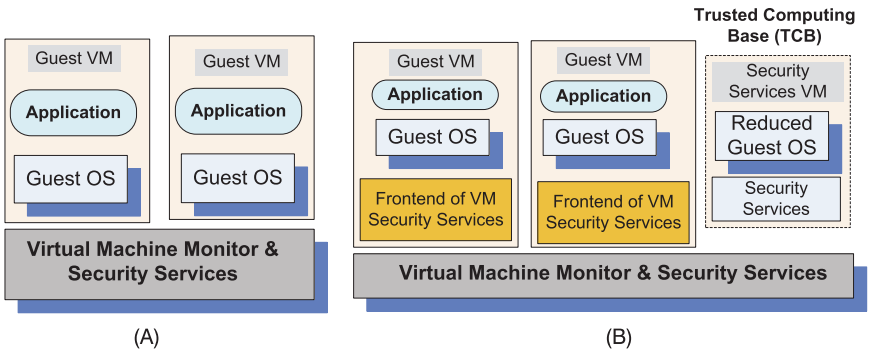

FIGURE 11.3

(A) Virtual security services provided by the hypervisor/Virtual Machine Monitor; (B) A dedicated security VM.

The analysis of Xen and *vBlades* in Sections 10.5 and 10.10 shows that the VM technology provides a stricter isolation of VMs from one another than the isolation of processes in a traditional operating system. Indeed, a hypervisor controls the execution of privileged operations and can thus enforce memory isolation as well as disk and network access.

Hypervisors are considerably less complex and better structured than traditional operating systems thus, in a better position to respond to security attacks. A major challenge is that a hypervisor sees only raw data regarding the state of a guest OS while 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 hypervisor has access to the state of all VMs operating on the same hardware. The state of a guest VM can be saved, restored, cloned, and encrypted by the hypervisor. Replication can ensure not only reliability but also support security, while cloning could be used to recognize a malicious application by testing it on a cloned system and observing if 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 VMs files moved to a dedicated VM and thus, protect it from attacks [549]. This solution is possible because inter-VM communication is faster than communication between two physical machines.

Sophisticated attackers are able to fingerprint VMs and avoid VM honey pots 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.

We expect to pay some price for the better security provided by virtualization. This price includes: (i) higher hardware costs because a virtual system requires more resources such as CPU cycles, memory, disk, and network bandwidth; (ii) the cost of developing hypervisors and modifying the host operating systems in case of paravirtualization; and (iii) the overhead of virtualization as the hypervisor is involved in privileged operations.

VM-based intrusion detection systems such as Livewire and Siren which exploit the three capabilities of a VM for intrusion detection, isolation, inspections, and interposition are surveyed in [549].

Resource isolation was examined in Section 8.10. Inspection means that the hypervisor has the ability to review the state of the guest VMs and interposition means that the hypervisor can trap and emulate the privileged instruction issued by the guest VMs. VM-based intrusion prevention systems such as, SVFS, NetTop, and IntroVirt, and surveys Terra, a VM-based trust computing platform are also discussed in [549]. Terra uses a *trusted hypervisor* to partition resources among VMs.

NIST security group distinguishes two groups of threats, hypervisor-based and VM-based.

There are several types of hypervisor-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 hypervisor.
2. VM side-channel attacks: malicious attack on one or more VMs by a rogue VM under the same hypervisor. Probable causes: (a) lack of proper isolation of inter-VM traffic due to misconfiguration of the virtual network residing in the hypervisor; (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.

There are also several types of 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, re-activation 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.

11.9 SECURITY OF VIRTUALIZATION

The complex relationship between virtualization and security has two distinct aspects: virtualization of security and security of virtualization [302]. In Chapter 10 we praised the virtues of virtualization. We also discussed two problems associated with virtual environments: (a) the negative effect on performance, due to the additional overhead; and (b) the need for more powerful systems to run multiple VMs. In this section we take a closer look at the security of virtualization.

The complete state of an operating system running under a VM is captured by the VM. The VM state can be saved in a file and then the file can be copied and shared. There are several useful implications of this important virtue of virtualization:

1. Supports the IaaS delivery model. An IaaS user selects an image matching the local application environment and then uploads and runs the application on the cloud using this image.
2. Increased reliability. An operating system with all the applications running under it can be replicated and switched to a hot standby in case of a system failure. Recall that a hot standby is a method to achieve redundancy. The primary and the backup systems, run simultaneously and have identical state information.
3. Straightforward mechanisms for implementing resource management policies. An OS and the applications running under it can be moved to another server to balance the load of a system. For

example, the load of lightly loaded servers can be moved to other servers and then lightly loaded servers can be switched off or placed in standby mode to reduce power consumption.

4. Improved intrusion detection. In a virtual environment a clone can look for known patterns in system activity and detect intrusion. The operator can switch a server to hot standby when suspicious events are detected.
5. Secure logging and intrusion protection. When implemented at the OS level intrusion detection can be disabled and logging can be modified by an intruder. When implemented at the hypervisor layer, the services cannot be disabled or modified. In addition, the hypervisor may be able to log only events of interest for a post-attack analysis.
6. More efficient and flexible software maintenance and testing. Virtualization allows the multitude of OS instances to share a small number of physical systems, instead of a large number of dedicated systems running under different operating systems, different versions of each OS, and different patches for each version.

Is there a price to pay for the benefits of virtualization outlined above? There is always the other side of a coin, so we should not be surprised that the answer to this question is a resounding Yes. In a 2005 paper [185] Garfinkel and Rosenblum argue that the serious implications of virtualization on system security cannot be ignored. This theme is revisited in 2008 by Price [410] who reaches similar conclusions.

A first group of undesirable effects of virtualization lead to a diminished ability of an organization to manage its systems and track their status. These undesirable effects are:

- The number of physical systems in the inventory of an organization is limited by cost, space, energy consumption, and human support. The explosion of the number of VMs is a fact of life; to create a VM one simply copies a file. The only limitation for the number of VMs is the amount of storage space available.
- There is also a qualitative side to the explosion of the number of VMs. Traditionally, organizations install and maintain the same version of system software. In a virtual environment such a uniformity cannot be enforced, the number of different operating systems, their versions, and the patch status of each version will be diverse and the diversity will tax the support team.
- One of the most critical problems posed by virtualization is related to the software lifecycle. The traditional assumption is that the software lifecycle is a straight line, hence the patch management is based on a monotonic forward progress. The virtual execution model *maps to a tree structure* rather than a line. Indeed, at any point in time multiple VM instances can be created and then each one of them can be updated, different patches installed, and so on. This problem has serious implication on security as we shall see shortly.

What are the direct implications of virtualization on security? A first question is how can the support team deal with the consequences of an attack in a virtual environment. Do we expect the infection with a computer virus or a worm to be less manageable in a virtual environment? The surprising answers to these questions is that an infection may last indefinitely.

Some of the infected VMs may be dormant at the time when the measures to clean up the systems are taken. Then, at a later time, the infected VMs wake up and infect other systems. The scenario can repeat itself and guarantee that infection will last indefinitely. This is in stark contrast with the manner an infection is treated in non-virtual environments. Once an infection is detected, the infected systems

are quarantined and then cleaned up; the systems will then behave normally until the next infection occurs.

A more general observation is that in a traditional computing environment a steady state can be reached. In this steady state all systems are brought up to a "desirable" state, whereas "undesirable" states, states when some of the systems are either infected by a virus or display an undesirable pattern of behavior, are only transient. The desirable state is reached by installing the latest system software version and then applying the latest patches to all systems.

A virtual environment may never reach such a steady state due to the lack of control. In a non-virtual environment the security can be compromised when an infected laptop is connected to the network protected by a firewall, or when a virus is brought in on a removable media. But, unlike a virtual environment, the system can still reach a steady state.

A side effect of the ability to record the complete state of a VM in a file is the possibility to roll back a VM. This opens wide the door for a new type of vulnerability caused by events recorded in the memory of an attacker. Two such situations are discussed in [185]. The first is that one-time passwords are transmitted in clear and the protection is guaranteed only if the attacker does not have the possibility to access passwords used in previous sessions.

An attacker can replay rolled back versions and access past sniffed passwords if a system runs the S/KEY password system. S/KEY is a password system based on Leslie Lamport's scheme. It is used by several operating systems including, Linux, OpenBSD, and NetBSD. The real password of the user is combined with a short set of characters and a counter that is decremented at each use to form a single-use password. The second situation is related to the requirement of some cryptographic protocols and even non-cryptographic protocols regarding the "freshness" of the random number source used for session keys and nonces. This situation occurs when a VM is rolled back to a state when a random number has been generated but not yet used.

A nonce is a random or pseudo-random number issued in an authentication protocol to ensure that old communications cannot be reused in replay attacks. For example, nonces are used to calculate the MD5 of the password for HTTP digest access authentication. Each time the authentication challenge response code is presented, the nonces are different, thus replay attacks are virtually impossible. This guarantees that an online order to Amazon or other online store cannot be replayed.

Even non-cryptographic use of random numbers may be affected by the rollback scenario. For example, the initial sequence number for a new TCP connection must be "fresh." The door to TCP hijacking is left open when the initial sequence number is not fresh.

Another undesirable effect of virtual environment affects trust. Recall from Section 11.4 that *trust is conditioned by the ability to guarantee the identity of entities involved.* Each computer system in a network has a unique physical, or MAC address. The uniqueness of the MAC address guarantees that an infected or a malicious system can be identified and then shut down, or denied network access. This process breaks down for virtual systems when VMs are created dynamically. Often, a random MAC address is assigned to a newly created VM to avoid name collision. The other effect discussed at length in Section 11.10 is that popular VM images are shared by many users.

The ability to guarantee confidentiality of sensitive data is yet another pillar of security affected by virtualization. Virtualization undermines the basic principle that time-sensitive data stored on any system should be reduced to a minimum. First, the owner has very limited control on where sensitive data is stored, it could be spread across many servers and may be left on some of them indefinitely.

A hypervisor records the state of a VM to be able to roll it back; this process allows an attacker to access sensitive data the owner attempted to destroy.

11.10 SECURITY RISKS POSED BY SHARED IMAGES

Even if we assume that a cloud service provider is trustworthy there are other sources of concern many users either ignore, or underestimate the danger they pose. One of them, especially critical for the IaaS cloud delivery model, is image sharing. For example, an AWS user 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 user, or for a less sophisticated one.

First, we 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, a process called *bundling*. The first of the three steps of 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 S3.

Two procedures, *ec2-bundle-image* and *ec2-bundle-volume*, are used for creation of an AMI. The first is used for images prepared as loopback files[3] when 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, a firewall configuration, and specify the region, as discussed in Section 2.3. Once the image is instantiated, the user is informed about the public DNS and the VM is available. A Linux system can be accessed using *ssh* at port 22, while the Remote Desktop at port 3389 is used for Windows.

A recent paper reports on the results of an analysis carried over a period of several months, from November 2010 to May 2011 of over five thousand AMIs available through the public catalog at Amazon [49]. Many analyzed images allowed a user to *undelete* files, recover credentials, private keys, or other types of sensitive information with little effort, using standard tools. The results of this study were shared with the 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 [49], here we only discuss the results of this analysis. 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 US, 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 of the malware targets Windows systems. Only

[3] A *loopback file system* (LOFS) is a virtual file system providing 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 CDROM 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.

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 for the two types of AMIs.

The *software vulnerability* audit revealed that 98% of the Windows AMIs (249 out of 253) and 58% (2005 out of 3432) Linux AMIs audited had critical vulnerabilities. The average number of vulnerabilities per AMI were 46 for Windows and 11 for Linux AMIs. Some of AMI images were rather old; 145, 38, and 2 Windows AMIs and 1197, 364, and 106 Linux AMIs were older than two, three, and four years, respectively. The tool used to detect vulnerabilities, the Nessus system, available from http://www.tenable.com/productus/nessus, classifies the vulnerabilities based on their severity in four groups, at levels zero to three. The audit reported only vulnerabilities of the highest severity level, e.g., remote code execution.

Three types of *security risks* are 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 login to the system. Some 100 passwords, 995 ssh keys, and 90 cases when both could be retrieved were identified.

To rent a Linux AMI a user must provide the public part of the her 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 her own public key from the image and can remotely login to 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 her own password. This backdoor is even wider open as one can extract the password hashes and then crack the passwords using a tool such as John the Riper, see http://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 attacks.

An attacker impersonates the agents at both ends of a communication channel in the *man-in-the middle* attack 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 her message with C's key, believing that she 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 with the public key B originally sent to A. When B receives the newly encrypted message, she believes it came from A.

When this script does not run, an attacker can use the *NMap* tool[4] to match the ssh keys discovered in the AMI images with the keys obtained with *NMap*. The study reports that the authors were able to identify more than 2100 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

[4] *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 to discover hosts and services in a network.

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* daemon 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 http://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 to decrypt and recover passwords stored by the Firefox browser, called *Trojan.Firepass*.

The creator of a shared AMI assumes some privacy risks. Her 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 which are not password protected. Then the malicious agent can start AMIs and run cloud applications at no cost to herself, as 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, the audit determined that the majority of ssh keys (54 out of 56) were not password protected. A *passphrase* is a sequence of words used to control access to a computer system. A passphrase is the analog of a password, but provides added security. For high security non-military applications NIST recommends an 80-bit strength passphrase. Therefore, 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 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 identify the domains contacted by the user.

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. The users should be aware that, when the HTTP protocol 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 then a malicious agent can redirect traffic from its original destination to her own system.

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 were 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.

11.11 SECURITY RISKS POSED BY A MANAGEMENT OS

We often hear that virtualization enhances security because a VM monitor or hypervisor is considerably smaller than an operating system. For example, the Xen hypervisor discussed in Section 10.5 has approximately 60 000 lines of code, one to two orders of magnitude fewer than a traditional operating system.[5]

A hypervisor supports a 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 hypervisor 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)[6] 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 Dom0, see Figure 11.4. 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 [116]; 11 attacks were attributed to problems in the guest OS caused by buffer overflow and 8 were denial of service attacks. Buffer overflow allows execution of arbitrary code in the privileged mode.

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 OS from secondary storage.
2. Allocate memory for the new VM and use foreign mapping to load the kernel to the new VM. The foreign mapping mechanism of Xen is used by Dom0 to map arbitrary memory frames of a VM into its page tables.
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.

[5]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.

[6]The TCB is defined as the totality of protection mechanisms within a computer system including hardware, firmware, and, software. The combination of all these elements is responsible for enforcing a security policy.

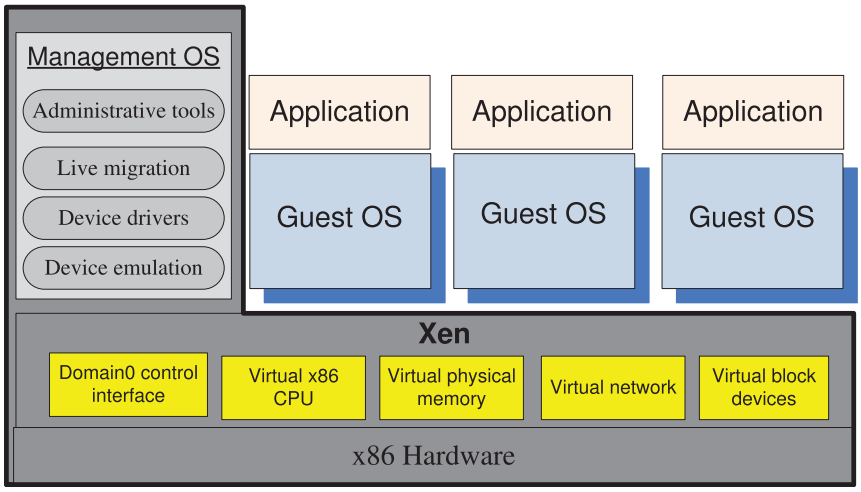

FIGURE 11.4

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 OS and applications running under it reside in a DomU.

A malicious Dom0 can play several nasty tricks at the time when it creates a DomU [302]:

- 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 OS 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 setup wrong virtual CPU registers.
- Refuse to release the foreign mapping and access the memory while the new VM is running.

We 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 frontend of such a driver is in the DomU and its backend in Dom0 and the two communicate via a ring in shared memory, see Section 10.5.

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. Note that later implementations of Xen offer the pass-through option.

Therefore, we have to ensure that run-time communication 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 10.5. 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 to use foreign mapping for sharing memory with a DomU, unless 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 [302]. 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. The guest software should have access to a secure secondary storage on a remote storage server and to the network interfaces when communicating with the user. A secure run-time system is also needed.

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 from 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 window. For example, hypercalls used by Dom0 for debugging or for the control of the IOMMU should be prohibited. The Input/Output Memory Management Unit (IOMMU) connects the 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.

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

New hypercalls are necessary to protect:

1. 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. The virtual CPU context should be decrypted and then an integrity check should be carried out when DomU is restored.
2. 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 Dome handles only encrypted pages of the VM. The hypervisor should calculate a hash of all the memory pages before they are saved by Dom0 to guarantee the integrity of the system. An address translation is necessary because a restored DomU may be allocated a different memory region [302].
3. 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 an increased overhead. Measurements reported in [302] show increases by a factor 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.

11.12 XOAR – BREAKING THE MONOLITHIC DESIGN OF THE TCB

Xoar is a modified version of Xen designed to boost system security [116]. The security model of Xoar assumes that the system is professionally managed and that a privileged access to the system is granted only to system administrators. The model also assumes that administrators have neither financial incentives, nor the desire to violate the user's trust. Security threats come from a guest VM which could attempt to violate the data integrity or the confidentiality of another guest VM on the same platform, or to exploit the code of the guest. Another source of threats are bugs in initialization code of the management VM.

Xoar is based on microkernel[7] design principles. Xoar modularity makes exposure to risk explicit and allows the guests to configure the access to services based on their needs. Modularity allows the designers of Xoar to reduce the size of the permanent footprint of the system and increase the level of security of critical components. Ability to record a secure audit log is another critical function of a hypervisor facilitated by a modular design. The design goals of Xoar are:

- Maintain the functionality provided by Xen.
- Ensure transparency with existing management and VM interfaces.
- Tight control of privileges. Each component should only have the privileges required by its function.
- Minimize the interfaces of all components to reduce the possibility that a component can be used by an attacker.
- Eliminate sharing. Make sharing explicit, whenever it cannot be eliminated, to allow meaningful logging and auditing.
- Reduce the opportunity of an attack targeting a system component by limiting the time window when the component runs.

These design principles aim to break the monolithic TCB design of a Xen-based system. Inevitably, this strategy has an impact on performance, but the implementation should attempt to keep the modularization overhead to a minimum.

A close analysis shows that booting the system is a complex activity, but the fairly large modules used during booting are no longer needed once the system is up and running. In Section 10.5 we have seen that XenStore is a critical system component, as it maintains the state of the system thus, it is a prime candidate for hardening. The ToolStack is only used for management functions and can only be loaded upon request.

The Xoar system has four types of components: permanent, self-destructing, restarted upon request, and restarted on timer, see Figure 11.5:

1. Permanent components. XenStore-State maintains all information regarding the state of the system.
2. Components used to boot the system; they self-destruct before any user VM is started. The two components discover the hardware configuration of the server including the PCI drivers and then boot the system:

[7] A microkernel (μ-kernel) supports only the basic functionality of an OS kernel including low-level address space management, thread management, and inter-process communication. Traditional OS components such as device drivers, protocol stacks, and file systems are removed from the microkernel and run in user space.

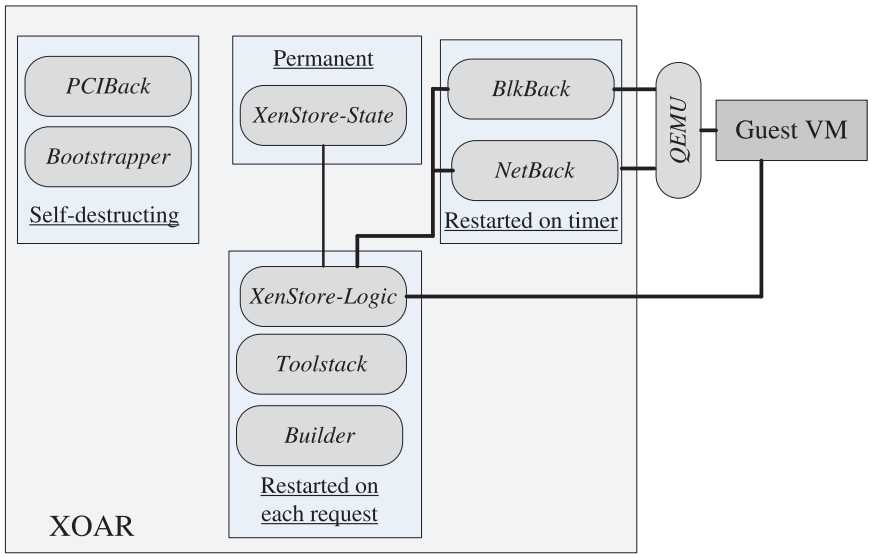

FIGURE 11.5

Xoar has nine classes of components of four types: permanent, self-destructing, restarted upon request, and restarted on timer. A guest VM is started using the Toolstack by the Builder and it is controlled by the XenStore-Logic. The devices used by the guest VM are emulated by the QEMU component.

- PCIBack – virtualizes access to PCI bus configuration.
- Bootstrapper – coordinates booting of the system.

3. Components restarted on each request:

   - XenStore-Logic
   - Toolstack – handles VM management requests, e.g., it requests the Builder to create a new guest VM in response to a user request.
   - Builder – initiates user VMs.

4. Components restarted on a timer: the two components export physical storage device drivers and the physical network driver to a guest VM.

   - BlkBack – exports physical storage device drivers using *udev*[8] rules.
   - NetBack – exports the physical network driver.

[8] *udev* is the device manager for the Linux kernel.

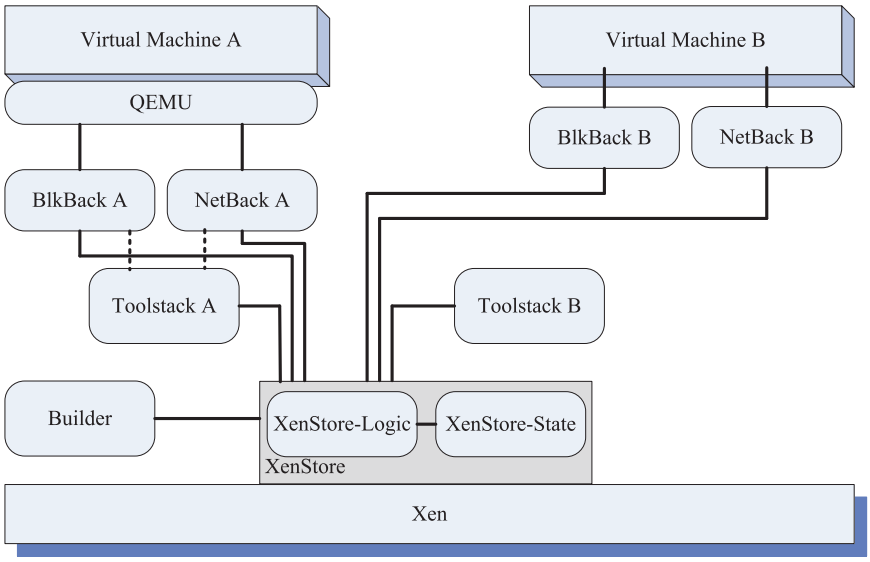

FIGURE 11.6

Component sharing between guest VM in Xoar. Two VM share only the *XenStore* components. Each one has a private version of the BlkBack, NetBack and Toolstack.

Another component, QEMU, is responsible for device emulation. Bootstrapper, PCIBack, and Builder are the most privileged components, but the first two are destroyed once Xoar is initialized. The Builder is very small, it consists of only 13 000 lines of code. XenStore is broken into two components, XenStore-Logic and XenStore-State. Access control checks are done by a small monitor module in XenStore-State. Guest VMs share only the Builder, XenStore-Logic, and XenStore-State, see Figure 11.6.

Users of Xoar are able to only share service VMs with guest VMs that they control; to do so they specify a tag on all of the devices of their hosted VMs. Auditing is more secure, whenever a VM is created, deleted, stopped, or restarted by Xoar the action is recorded in an append-only database on a different server accessible via a secure channel.

Rebooting provides the means to ensure that a VM is in a known good state. To reduce the overhead and the increased startup time demanded by a reboot, Xoar uses *snapshots* instead of rebooting. The service VM snapshots itself when it is ready to service a request. Similarly, snapshots of all components are taken immediately after their initialization and before they start interacting with other services or guest VMs. Snapshots are implemented using a copy-on-write mechanism[9] to preserve any page about to be modified.

[9]Copy-on-write (COW) is used by virtual memory operating systems to minimize the overhead of copying the virtual memory of a process when a process creates a copy of itself. Then the pages in memory that might be modified by the process or by its copy are marked as COW. When one process modifies the memory, the operating system's kernel intercepts the operation and copies the memory so that changes in one process's memory are not visible to the other.

11.13 A TRUSTED HYPERVISOR

After the discussion of Xoar we briefly analyze the design of a trusted hypervisor called *Terra* [184]. The novel ideas of this design are:

- A trusted hypervisor should support not only traditional operating systems, by exporting the hardware abstraction for open-box platforms, but also the abstractions for closed-box platforms discussed in Section 11.7. Note that the VM abstraction for a closed-box platform does not allow the contents of the system to be either manipulated or inspected by the platform owner.
- An application should be allowed to build its software stack based on its needs. Applications requiring a very high level of security, e.g., financial applications and electronic voting systems should run under a very thin OS supporting only the functionality required by the application and the ability to boot. At the other end of the spectrum are applications demanding low assurance, but a rich set of OS features. Such applications need a commodity operating system. *Information assurance* (IA) means to manage the risks related to the use, processing, storage, and transmission of information, as well as protecting the systems and processes used for those purposes. IA implies protection of the integrity, availability, authenticity, non-repudiation and confidentiality of the application data.
- Support additional capabilities to enhance system assurance:
  - Provide trusted paths from a user to an application. We have seen in Section 11.7 that such a path allows a human user to determine with certainty the identity of the VM it is interacting with and, at the same time, allows the VM to verify the identity of the human user.
  - Support attestation, the ability of an application running in a closed-box to gain trust from a remote party, by cryptographically identifying itself.
  - Provide air-tight isolation guarantees for the hypervisor by denying the platform administrator the root access.

The management VM is selected by the owner of the platform but makes a distinction between the *platform owner* and a *platform user*. The management VM formulates limits for the number of guest VMs running on the platform, denies access to the guest VM deemed unsuitable to run, grants access to I/O devices to running VMs and limits their CPU, memory, and disk usage.

Guest VMs expose a raw hardware interface including virtual network interfaces to virtual devices. The trusted hypervisor runs at the highest privilege level and it is secure even from the actions of the platform owner; it provides application developers with the semantics of a closed-box platform.

A significant challenge to the security of a trusted hypervisor comes from the device drivers used by different VMs running on the platform. Device drivers are large or very large software components, especially the drivers for high-end wireless cards and video cards. There is also a large variety of such drivers, many hastily written to accommodate new hardware features.

Typically, the device drivers are the lowest quality software components found in the kernel of an operating system thus, they pose the highest security risks. To protect a trusted hypervisor, the device drivers should not be allowed to access sensitive information and their memory access should be limited by different hardware protection mechanisms. Malicious I/O devices can use different hardware capabilities such as DMA to modify the kernel.

11.14 MOBILE DEVICES AND CLOUD SECURITY

Mobile devices are an integral part of the cloud ecosystem, mobile applications use cloud services to access and store data or to carry out a multitude of computational tasks. Security challenges for mobile devices common to all computer and communication systems include: (i) Confidentiality – ensure that transmitted and stored data cannot be read by unauthorized parties; (ii) Integrity – detect intentional or unintentional changes to transmitted and stored data; (iii) Availability – ensure that users can access cloud resources whenever needed; and (iv) Non-repudiation – the ability to ensure that a party to a contract cannot deny the sending of a message that they originated.

The technology stack of a mobile device consists of the hardware, the firmware, the operating system, and the applications. The separation between the firmware and the hardware of a mobile device is blurred. A baseband processor is used solely for telephony services involving data transfers over cellular networks operating outside the control of the mobile OS which runs on the application processor. Security-specific hardware and firmware store encryption keys, certificates, credentials, and other sensitive information on some mobile devices.

The nature of mobile devices places them at higher exposure to threats than stationary ones. Mobile devices are designed to easily install applications, to use third-party applications from application stores, and to communicate with computer clouds via often untrusted cellular and WiFi networks. Mobile devices interact frequently with other systems to exchange data and often use untrusted content.

Mobile devices often require a short authentication passcode and may not support strong storage encryption. Location services increase the risk of targeted attacks. Potential attackers are able to determine user's location, correlate the location with information from other sources on the individuals the user associates with, and infer other sensitive information.

Special precautions must then be taken due to exposure to the unique security threats affecting mobile devices, including:

1. Mobile malware.
2. Stolen data due to loss, theft, or disposal.
3. Unauthorized access.
4. Electronic eavesdropping.
5. Electronic tracking.
6. Access to data by third party applications.

Some of these threats can propagate to the cloud infrastructure a mobile device is connected to. For example, files stored on the mobile devices subject to ransomeware and encrypted by a malicious intruder can migrate to the backup stored on the cloud. The risks posed to the cloud infrastructure by mobile devices are centered around data leakage and compromise. Such security risks are due to a set of reasons including:

- Loss of the mobile device, lock screen protection, enabling smudge attacks and other causes leading to mobile access control. A smudge attack is a method to discern the password pattern of a touchscreen device such as a cell phone or tablet computer.
- Lack of confidentiality protection for data in transit in unsafe or untrusted WiFi or cellular networks.
- Unmatched firmware or software including operating system and application software bypassing the security architecture, e.g., rooted/jailbroken devices.
- Malicious mobile applications bypassing access control mechanisms.

- Misuse or misconfiguration of location services, such as GPS.
- Acceptance of fake mobility management profiles.

An in-depth discussion of the Enterprise Mobile Management (EMM) lists different EMM services including the Mobile Device Management (MDM) and the Mobile Application Management (MAM) and suggests a number of functional and security capabilities of the system [179]. Some of these policies and mechanisms should also be applied to mobile devices connected to computer cloud:

1. Use device encryption, application-level encryption, and remote wipe capabilities to protect storage.
2. Use Transport Layer Security (TLS) for all communication channels.
3. Isolate user-level applications from each other to prevent data leakage between applications using sandboxing.
4. Use device integrity checks for boot validation, verified application and OS updates.
5. Use auditing and logging.
6. Enforce authentication of the device owner.
7. Automatic, regular device integrity and compliance checks for threats and compliance.
8. Automated alerts for policy violations.

A system for Microsoft Outlook mobile application requires individuals who wish to participate in a managed scenario to download the Microsoft Community Portal application and input the required information including local authentication to the mobile OS via a lockscreen and the encryption capabilities provided by the mobile OS to protect data on the device. The cloud MDM portal discussed in [179] is available to administrators through a web interface.

11.15 FURTHER READINGS

The Cloud Security Alliance (CSA) is an organization with more than 100 corporate members. It aims to address all aspects of cloud security and serve as a cloud security standards incubator. The reports, available from the web site of the organization are periodically updated; the original report was published in 2009 [122] and subsequent reports followed [123] and [124]. An open security architecture is presented in [382].

A seminal paper [185] on the negative implications of virtualization on system security "When virtual is harder than real: security challenges in VM based computing environments" by Garfinkel and Rosenblum was published in 2005, followed by another one which reaches similar conclusions, [410]. Risk and trust are analyzed in [153], [252], and [317]. Cloud security is also discussed in [339], [468], and [474]. Managing cloud information leakage is the topic of [519].

A 2010 paper, [207] presents a taxonomy of attacks on computer clouds and [138] covers the management of security services lifecycle. Security issues vary depending on the cloud model as discussed in [377]. The privacy impact in cloud computing is the topic of [478]. A 2011 book [523] gives a comprehensive look at cloud security. Privacy and protection of personal data in the EU is discussed in a document available at http://ec.europa.eu/justice/policies/privacy.

The paper [40] analyzes the inadequacies of current risk controls for the cloud. Intercloud security is the theme of [63]. Secure collaborations are discussed in [66]. The paper [303] presents an approach for secure VM execution under untrusted management OS. The social impact of privacy in cloud computing is analyzed in [166]. An anonymous access control scheme is presented in [257].

An empirical study into the security exposure to hosts of hostile virtualized environments can be found at http://taviso.decsystem.org/virtsec.pdf. A model-based security testing approach for cloud computing is presented in [544]. Cold boot attacks on encryption keys are discussed in [217]. Cloud security concerns and mobile device security are covered in [370] and [371], respectively. [405] introduces an encryption system for query processing and [439] discusses a pragmatic security discipline.

11.16 EXERCISES AND PROBLEMS

Problem 1. Identify the main security threats for the SaaS cloud delivery model on a public cloud. Discuss the different aspects of these threats on a public cloud as compared to the threats posed to similar services provided by a traditional service-oriented architecture running on a private infrastructure.

Problem 2. Analyze how the six attack surfaces discussed in Section 11.2 and illustrated in Figure 11.1 apply to the SaaS, PaaS and IaaS cloud delivery models.

Problem 3. Analyze Amazon privacy policies and design a service level agreement you would sign on if you were to process confidential data using AWS.

Problem 4. Analyze the implications of the lack of trusted paths in commodity operating systems and give one or more examples showing the effects of this deficiency. Analyze the implications of the two-level security model of commodity operating systems.

Problem 5. Compare the benefits and the potential problems due to virtualization on public, private, and hybrid clouds.

Problem 6. Read [49] and discuss the measures taken by Amazon to address the problems posed by shared images available from AWS. Would it be useful to have a cloud service to analyze images and sign them before being listed and made available to the general public?

Problem 7. Analyze the risks posed by foreign mapping and the solution adopted by Xoar. What is the security risk posed by XenStore?

Problem 8. Read [116] and discuss the performance of the system. What obstacles to its adoption by the providers of IaaS services can you foresee?

Problem 9. Discuss the impact of international agreements regarding privacy laws on cloud computing.

Problem 10. Propagation of the malware in the Internet has similarities with the propagation of an infectious disease. Discuss the three models for the propagation of an infectious disease in a finite population, *SI, SIR*, and *SIS*. Justify the formulas describing the dynamics of the system for each model. Hint: read [76,161] and [267].

This page intentionally left blank