CHAPTER 12

# OPERATING SYSTEM SECURITY

**Learning Objectives**

After studying this chapter, you should be able to:

- List the steps needed in the process of securing a system.
- Detail the need for planning system security.
- List the basic steps used to secure the base operating system.
- List the additional steps needed to secure key applications.
- List steps needed to maintain security.
- List some specific aspects of securing Unix/Linux systems.
- List some specific aspects of securing Windows systems.
- List steps needed to maintain security in virtualized systems.

Computer client and server systems are central components of the IT infrastructure for most organizations. The client systems provide access to organizational data and applications, supported by the servers housing those data and applications. However, given that most large software systems will almost certainly have a number of security weaknesses, as we discussed in Chapter 6 and in the previous two chapters, it is currently necessary to manage the installation and continuing operation of these systems to provide appropriate levels of security despite the expected presence of these vulnerabilities. In some circumstances, we may be able to use trusted computing systems designed and evaluated to provide security by design. We will examine some of these possibilities in Chapter 27.

In this chapter, we discuss how to provide systems security as a hardening process that includes planning, installation, configuration, update, and maintenance of the operating system and the key applications in use, following the general approach detailed in NIST SP 800-123 (*Guide to General Server Security*, July 2008). We consider this process for the operating system, and then key applications in general, then discuss some specific aspects in relation to Linux and Windows systems in particular. We conclude with a discussion on securing virtualized systems, where multiple virtual machines may execute on the one physical system.

We view a system as having a number of layers, with the physical hardware at the bottom; the base operating system above including privileged kernel code, APIs, and services; and finally user applications and utilities in the top layer, as shown in Figure 12.1. This figure also shows the presence of BIOS and possibly other code that

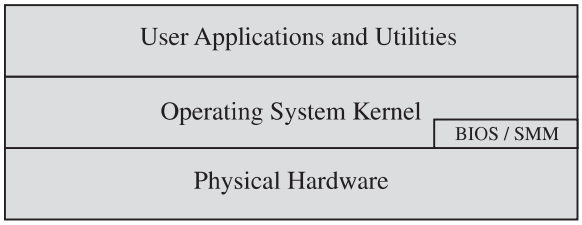



**Figure 12.1 Operating System Security Layers**

is external to, and largely not visible from, the operating system kernel, but is used when booting the system or to support low-level hardware control. Each of these layers of code needs appropriate **hardening** measures in place to provide appropriate security services. And each layer is vulnerable to attack from below, should the lower layers not also be secured appropriately.

A number of reports note the use of a small number of basic hardening measures can prevent a large proportion of the attacks seen in recent years. Since 2010, the Australian Signals Directorate (ASD) list of the "Top 35 Mitigation Strategies" notes that implementing just the top four of these strategies would have prevented at least 85% of the targeted cyber intrusions investigated by ASD. Hence, since 2013 these top four strategies are mandatory for all Australian government agencies. These top four strategies are as follows:

1. White-list approved applications.
2. Patch third-party applications.
3. Patch operating system vulnerabilities and use the latest versions.
4. Restrict administrative privileges.

Collectively these assist in creating a defence-in-depth system. We discuss all four of these strategies, and many others in the ASD list, in this chapter. Note these strategies largely align with those in the "20 Critical Controls" developed by DHS, NSA, the Department of Energy, SANS, and others in the United States.

## 12.1 INTRODUCTION TO OPERATING SYSTEM SECURITY

As we noted above, computer client and server systems are central components of the IT infrastructure for most organizations, may hold critical data and applications, and are a necessary tool for the function of an organization. Accordingly, we need to be aware of the expected presence of vulnerabilities in operating systems and applications as distributed, and the existence of worms scanning for such vulnerabilities at high rates, such as those we discussed in Section 6.3. Thus, it is quite possible for a system to be compromised during the installation process, before it can install the latest patches or implement other hardening measures. Hence, building and deploying a system should be a planned process designed to counter such a threat, and to maintain security during its operational lifetime.

NIST SP 800-123 states that this process must:

- Assess risks and plan the system deployment.
- Secure the underlying operating system and then the key applications.
- Ensure any critical content is secured.
- Ensure appropriate network protection mechanisms are used.
- Ensure appropriate processes are used to maintain security.

While we addressed the selection of network protection mechanisms in Chapter 9, we will examine the other items in the rest of this chapter.

## 12.2 SYSTEM SECURITY PLANNING

The first step in deploying new systems is planning. Careful planning will help to ensure that the new system is as secure as possible, and complies with any necessary policies. This planning should be informed by a wider security assessment of the organization, since every organization has distinct security requirements and concerns. We will discuss this wider planning process in Chapters 14 and 15.

The aim of the specific system installation planning process is to maximize security while minimizing costs. Wide experience shows that it is much more difficult and expensive to "retro-fit" security at a later time, than it is to plan and provide it during the initial deployment process. This planning process needs to determine the security requirements for the system, its applications and data, and of its users. This then guides the selection of appropriate software for the operating system and applications, and provides guidance on appropriate user configuration and access control settings. It also guides the selection of other hardening measures required. The plan also needs to identify appropriate personnel to install and manage the system, noting the skills required and any training needed.

NIST SP 800-123 provides a list of items that should be considered during the system security planning process. While its focus is on secure server deployment, much of the list applies equally well to client system design. This list includes consideration of:

- The purpose of the system, the type of information stored, the applications and services provided, and their security requirements.
- The categories of users of the system, the privileges they have, and the types of information they can access.
- How the users are authenticated.
- How access to the information stored on the system is managed.
- What access the system has to information stored on other hosts, such as file or database servers, and how this is managed.
- Who will administer the system, and how they will manage the system (via local or remote access).
- Any additional security measures required on the system, including the use of host firewalls, anti-virus or other malware protection mechanisms, and logging.

## 12.3 OPERATING SYSTEMS HARDENING

The first critical step in securing a system is to secure the base operating system upon which all other applications and services rely. A good security foundation needs a properly installed, patched, and configured operating system. Unfortunately, the default configuration for many operating systems often maximizes ease of use and functionality, rather than security. Further, since every organization has its own security needs, the appropriate security profile, and hence configuration, will also differ. What is required for a particular system should be identified during the planning phase, as we have just discussed.

While the details of how to secure each specific operating system differ, the broad approach is similar. Appropriate security configuration guides and checklists exist for most common operating systems, and these should be consulted, though always informed by the specific needs of each organization and their systems. In some cases, automated tools may be available to further assist in securing the system configuration.

NIST SP 800-123 suggests the following basic steps that should be used to secure an operating system:

- Install and patch the operating system.
- Harden and configure the operating system to adequately address the identified security needs of the system by:
  - Removing unnecessary services, applications, and protocols.
  - Configuring users, groups, and permissions.
  - Configuring resource controls.
- Install and configure additional security controls, such as anti-virus, host-based firewalls, and intrusion detection systems (IDS), if needed.
- Test the security of the basic operating system to ensure that the steps taken adequately address its security needs.

### Operating System Installation: Initial Setup and Patching

System security begins with the installation of the operating system. As we have already noted, a network connected, unpatched system, is vulnerable to exploit during its installation or continued use. Hence, it is important that the system not be exposed while in this vulnerable state. Ideally, new systems should be constructed on a protected network. This may be a completely isolated network, with the operating system image and all available patches transferred to it using removable media such as DVDs or USB drives. Given the existence of malware that can propagate using removable media, as we discussed in Chapter 6, care is needed to ensure the media used here is not so infected. Alternatively, a network with severely restricted access to the wider Internet may be used. Ideally, it should have no inbound access, and have outbound access only to the key sites needed for the system installation and patching process. In either case, the full installation and hardening process should occur before the system is deployed to its intended, more accessible, and hence vulnerable, location.

The initial installation should install the minimum necessary for the desired system, with additional software packages included only if they are required for the function of the system. We explore the rationale for minimizing the number of packages on the system shortly.

The overall boot process must also be secured. This may require adjusting options on, or specifying a password required for changes to, the BIOS code used when the system initially boots. It may also require limiting from which media the system is normally permitted to boot. This is necessary to prevent an attacker from changing the boot process to install a covert hypervisor, such as we discussed in Section 6.8, or to just boot a system of their choice from external media in order to bypass the normal system access controls on locally stored data. The use of a cryptographic file system may also be used to address this threat, as we will note later.

Care is also required with the selection and installation of any additional device driver code, since this executes with full kernel level privileges, but is often supplied by a third party. The integrity and source of such driver code must be carefully validated given the high level of trust it has. A malicious driver can potentially bypass many security controls to install malware. This was done in both the Blue Pill demonstration rootkit, which we discussed in Section 6.8, and the Stuxnet worm, which we described in Section 6.3.

Given the continuing discovery of software and other vulnerabilities for commonly used operating systems and applications, it is critical that the system be kept as up to date as possible, with all critical security related **patches** installed. Indeed, doing this addresses one of the top four key ASD mitigation strategies we listed previously. Nearly, all commonly used systems now provide utilities that can automatically download and install security updates. These tools should be configured and used to minimize the time any system is vulnerable to weaknesses for which patches are available.

On change-controlled systems, there can be a perception that running automatic updates may be detrimental, as they may on rare but significant occasions, introduce instability. However, ASD notes, that the delay in testing patches can leave systems vulnerable to compromise, and that they believe automatic update is preferable. For systems on which availability and uptime are of paramount importance, you may need to stage and validate all patches on test systems before deploying them in production. However, this process should be as timely as possible.

### Remove Unnecessary Services, Application, and Protocols

Because any of the software packages running on a system may contain software vulnerabilities, clearly if fewer software packages are available to run, then the risk is reduced. There is clearly a balance between usability, providing all software that may be required at some time, with security, and a desire to limit the amount of software installed. The range of services, applications, and protocols required will vary widely between organizations, and indeed between systems within an organization. The system planning process should identify what is actually required for a given system, so a suitable level of functionality is provided, while eliminating software that is not required to improve security.

The default configuration for most distributed systems is set to maximize ease of use and functionality, rather than security. When performing the initial installation, the supplied defaults should not be used, but rather the installation should be customized so only the required packages are installed. If additional packages are needed later, they can be installed when they are required. NIST SP 800-123 and many of the security hardening guides provide lists of services, applications, and protocols that should not be installed if not required.

NIST SP 800-123 also states a strong preference for not installing unwanted software, rather than installing then later removing or disabling it. It argues this preference because they note that many uninstall scripts fail to completely remove all components of a package. They also note that disabling a service means that while it is not available as an initial point of attack, should an attacker succeed in gaining some access to a system, then disabled software could be re-enabled and used to further compromise a system. It is better for security if unwanted software is not installed, and thus not available for use at all.

### Configure Users, Groups, and Authentication

Not all users with access to a system will have the same access to all data and resources on that system. All modern operating systems implement **access controls** to data and resources, as we discussed in Chapter 4. Nearly, all provide some form of discretionary access controls. Some systems may provide role-based or mandatory access control mechanisms as well.

The system planning process should consider the categories of users on the system, the privileges they have, the types of information they can access, and how and where they are defined and authenticated. Some users will have elevated privileges to administer the system; others will be normal users, sharing appropriate access to files and other data as required; and there may even be guest accounts with very limited access. The last of the four key ASD mitigation strategies is to restrict elevated privileges to only those users that require them. Further, it is highly desirable that such users only access elevated privileges when needed to perform some task that requires them, and to otherwise access the system as a normal user. This improves security by providing a smaller window of opportunity for an attacker to exploit the actions of such privileged users. Some operating systems provide special tools or access mechanisms to assist administrative users to elevate their privileges only when necessary, and to appropriately log these actions.

One key decision is whether the users, the groups they belong to, and their authentication methods are specified locally on the system or will use a centralized authentication server. Whichever is chosen, the appropriate details are now configured on the system.

Also at this stage, any default accounts included as part of the system installation should be secured. Those which are not required should be either removed or at least disabled. System accounts that manage services on the system should be set so they cannot be used for interactive logins. And any passwords installed by default should be changed to new values with appropriate security.

Any policy that applies to authentication credentials, and especially to password security, is also configured. This includes details of which authentication methods are accepted for different methods of account access. And it includes details of the required length, complexity, and age allowed for passwords. We discussed some of these issues in Chapter 3.

### Configure Resource Controls

Once the users and their associated groups are defined, appropriate **permissions** can be set on data and resources to match the specified policy. This may be to limit which users can execute some programs, especially those that modify the system state. Or it may be to limit which users can read or write data in certain directory trees. Many of the security hardening guides provide lists of recommended changes to the default access configuration to improve security.

### Install Additional Security Controls

Further security improvement may be possible by installing and configuring additional security tools such as antivirus software, host-based firewalls, IDS or IPS software, or application white-listing. Some of these may be supplied as part of the

operating systems installation, but not configured and enabled by default. Others are third-party products that are acquired and used.

Given the widespread prevalence of malware, as we discussed in Chapter 6, appropriate anti-virus (which as noted addresses a wide range of malware types) is a critical security component on many systems. Anti-virus products have traditionally been used on Windows systems, since their high use made them a preferred target for attackers. However, the growth in other platforms, particularly smartphones, has led to more malware being developed for them. Hence, appropriate anti-virus products should be considered for any system as part of its security profile.

Host-based firewalls, IDS, and IPS software also may improve security by limiting remote network access to services on the system. If remote access to a service is not required, though some local access is, then such restrictions help secure such services from remote exploit by an attacker. Firewalls are traditionally configured to limit access by port or protocol, from some or all external systems. Some may also be configured to allow access from or to specific programs on the systems, to further restrict the points of attack, and to prevent an attacker installing and accessing their own malware. IDS and IPS software may include additional mechanisms such as traffic monitoring, or file integrity checking to identify and even respond to some types of attack.

Another additional control is to white-list applications. This limits the programs that can execute on the system to just those in an explicit list. Such a tool can prevent an attacker installing and running their own malware, and is the first of the four key ASD mitigation strategies. While this will improve security, it functions best in an environment with a predictable set of applications that users require. Any change in software usage would require a change in the configuration, which may result in increased IT support demands. Not all organizations or all systems will be sufficiently predictable to suit this type of control.

### Test the System Security

The final step in the process of initially securing the base operating system is security testing. The goal is to ensure that the previous security configuration steps are correctly implemented, and to identify any possible vulnerabilities that must be corrected or managed.

Suitable checklists are included in many security hardening guides. There are also programs specifically designed to review a system to ensure that a system meets the basic security requirements, and to scan for known vulnerabilities and poor configuration practices. This should be done following the initial hardening of the system, and then repeated periodically as part of the security maintenance process.

## 12.4 APPLICATION SECURITY

Once the base operating system is installed and appropriately secured, the required services and applications must next be installed and configured. The steps for this very much mirror the list already given in the previous section. The concern, as with the base operating system, is to only install software on the system that is required to

meet its desired functionality, in order to reduce the number of places vulnerabilities may be found. On client systems, software such as Java, PDF viewers, Flash, Web browsers, and Microsoft Office are known targets and need to be secured. On server systems, software that provides remote access or service, including Web, database, and file access servers, is of particular concern, since an attacker may be able to exploit this to gain remote access to the system.

Each selected service or application must be installed, configured, and then patched to the most recent supported secure version appropriate for the system. This may be from additional packages provided with the operating system distribution, or from a separate third-party package. As with the base operating system, utilizing an isolated, secure build network is preferred.

### Application Configuration

Any application specific configuration is then performed. This may include creating and specifying appropriate data storage areas for the application, and making appropriate changes to the application or service default configuration details.

Some applications or services may include default data, scripts, or user accounts. These should be reviewed, and only retained if required, and suitably secured. A well-known example of this is found with Web servers, which often include a number of example scripts, quite a few of which are known to be insecure. These should not be used as supplied, but should be removed unless needed and secured.

As part of the configuration process, careful consideration should be given to the access rights granted to the application. Again, this is of particular concern with remotely accessed services, such as Web and file transfer services. The server application should not be granted the right to modify files, unless that function is specifically required. A very common configuration fault seen with Web and file transfer servers is for all the files supplied by the service to be owned by the same "user" account that the server executes as. The consequence is that any attacker able to exploit some vulnerability in either the server software or a script executed by the server may be able to modify any of these files. The large number of "Web defacement" attacks is clear evidence of this type of insecure configuration. Much of the risk from this form of attack is reduced by ensuring that most of the files can only be read, but not written, by the server. Only those files that need to be modified, to store uploaded form data for example, or logging details, should be writeable by the server. Instead the files should mostly be owned and modified by the users on the system who are responsible for maintaining the information.

### Encryption Technology

Encryption is a key enabling technology that may be used to secure data both in transit and when stored, as we discussed in Chapter 2 and in Parts Four and Five. If such technologies are required for the system, then they must be configured, and appropriate cryptographic keys created, signed, and secured.

If secure network services are provided, most likely using either TLS or IPsec, then suitable public and private keys must be generated for each of them. Then X.509 certificates are created and signed by a suitable certificate authority, linking each service identity with the public key in use, as we will discuss in Section 23.2. If secure

remote access is provided using Secure Shell (SSH), then appropriate server, and possibly client keys, must be created.

Cryptographic file systems are another use of encryption. If desired, then these must be created and secured with suitable keys.

## 12.5 SECURITY MAINTENANCE

Once the system is appropriately built, secured, and deployed, the process of maintaining security is continuous. This results from the constantly changing environment, the discovery of new vulnerabilities, and hence exposure to new threats. NIST SP 800-123 suggests that this process of security maintenance includes the following additional steps:

- Monitoring and analyzing logging information
- Performing regular backups
- Recovering from security compromises
- Regularly testing system security
- Using appropriate software maintenance processes to patch and update all critical software, and to monitor and revise configuration as needed

We have already noted the need to configure automatic **patching** and update where possible, or to have a timely process to manually test and install patches on high availability systems, and that the system should be regularly tested using checklist or automated tools where possible. We will discuss the process of incident response in Section 17.4. We now consider the critical logging and backup procedures.

### Logging

NIST SP 800-123 notes that "logging is a cornerstone of a sound security posture." **Logging** is a reactive control that can only inform you about bad things that have already happened. But effective logging helps ensure that in the event of a system breach or failure, system administrators can more quickly and accurately identify what happened and thus most effectively focus their remediation and recovery efforts. The key is to ensure you capture the correct data in the logs, and are then able to appropriately monitor and analyze this data. Logging information can be generated by the system, network, and applications. The range of logging data acquired should be determined during the system planning stage, as it depends on the security requirements and information sensitivity of the server.

Logging can generate significant volumes of information. It is important that sufficient space is allocated for them. A suitable automatic log rotation and archive system should also be configured to assist in managing the overall size of the logging information.

Manual analysis of logs is tedious and is not a reliable means of detecting adverse events. Rather, some form of automated analysis is preferred, as it is more likely to identify abnormal activity. Intrusion Detection Systems, such as those we discuss in Chapter 8, perform such automated analysis.

We will discuss the process of logging further in Chapter 18.

### Data Backup and Archive

Performing regular backups of data on a system is another critical control that assists with maintaining the integrity of the system and user data. There are many reasons why data can be lost from a system, including hardware or software failures, or accidental or deliberate corruption. There may also be legal or operational requirements for the retention of data. **Backup** is the process of making copies of data at regular intervals, allowing the recovery of lost or corrupted data over relatively short time periods of a few hours to some weeks. **Archive** is the process of retaining copies of data over extended periods of time, being months or years, in order to meet legal and operational requirements to access past data. These processes are often linked and managed together, although they do address distinct needs.

The needs and policy relating to backup and archive should be determined during the system planning stage. Key decisions include whether the backup copies are kept online or offline, and whether copies are stored locally or transported to a remote site. The trade-offs include ease of implementation and cost versus greater security and robustness against different threats.

A good example of the consequences of poor choices here was seen in the attack on an Australian hosting provider in early 2011. The attackers destroyed not only the live copies of thousands of customer's sites, but also all of the online backup copies. As a result, many customers who had not kept their own backup copies lost all of their site content and data, with serious consequences for many of them, and for the hosting provider as well. In other examples, many organizations that only retained onsite backups have lost all their data as a result of fire or flooding in their IT center. These risks must be appropriately evaluated.

## 12.6 LINUX/UNIX SECURITY

Having discussed the process of enhancing security in operating systems through careful installation, configuration, and management, we now consider some specific aspects of this process as it relates to Unix and Linux systems. Beyond the general guidance in this section, we will provide a more detailed discussion of Linux security mechanisms in Chapter 25.

There are a large range of resources available to assist **administrators** of these systems, including many texts, for example [NEME10], online resources such as the "Linux Documentation Project," and specific system hardening guides such as those provided by the "NSA—Security Configuration Guides." These resources should be used as part of the system security planning process in order to incorporate procedures appropriate to the security requirements identified for the system.

### Patch Management

Ensuring that system and application code is kept up to date with security patches is a widely recognized and critical control for maintaining security.

Modern Unix and Linux distributions typically include tools for automatically downloading and installing software updates, including security updates, which can minimize the time a system is vulnerable to known vulnerabilities for which patches

exist. For example, Red Hat, Fedora, and CentOS include `up2date` or `yum`; SuSE includes `yast`; and Debian uses `apt-get`, though you must run it as a cron job for automatic updates. It is important to configure whichever update tool is provided on the distribution in use, to install at least critical security patches in a timely manner.

As noted earlier, high availability systems that do not run automatic updates, as they may possibly introduce instability, should validate all patches on test systems before deploying them to production systems.

### Application and Service Configuration

Configuration of applications and services on Unix and Linux systems is most commonly implemented using separate text files for each application and service. System-wide configuration details are generally located either in the "`/etc`" directory or in the installation tree for a specific application. Where appropriate, individual user configurations that can override the system defaults are located in hidden "dot" files in each user's home directory. The name, format, and usage of these files are very much dependent on the particular system version and applications in use. Hence, the systems administrators responsible for the secure configuration of such a system must be suitably trained and familiar with them.

Traditionally, these files were individually edited using a text editor, with any changes made taking effect either when the system was next rebooted or when the relevant process was sent a signal indicating that it should reload its configuration settings. Current systems often provide a GUI interface to these configuration files to ease management for novice administrators. Using such a manager may be appropriate for small sites with a limited number of systems. Organizations with larger numbers of systems may instead employ some form of centralized management, with a central repository of critical configuration files that can be automatically customized and distributed to the systems they manage.

The most important changes needed to improve system security are to disable services, especially remotely accessible services, and applications, that are not required, and to then ensure that applications and services that are needed are appropriately configured, following the relevant security guidance for each. We will provide further details on this in Section 25.5.

### Users, Groups, and Permissions

As we describe in Sections 4.4 and 25.3, Unix and Linux systems implement discretionary access control to all file system resources. These include not only files and directories but also devices, processes, memory, and indeed most system resources. Access is specified as granting read, write, and execute **permissions** to each of owner, group, and others, for each resource, as shown in Figure 4.5. These are set using the `chmod` command. Some systems also support extended file attributes with access control lists that provide more flexibility, by specifying these permissions for each entry in a list of users and groups. These extended access rights are typically set and displayed using the `getfacl` and `setfacl` commands. These commands can also be used to specify set user or set group permissions on the resource.

Information on user accounts and group membership are traditionally stored in the `/etc/passwd` and `/etc/group` files, though modern systems also have the

ability to import these details from external repositories queried using LDAP or NIS for example. These sources of information, and indeed of any associated authentication credentials, are specified in the PAM (pluggable authentication module) configuration for the system, often using text files in the `/etc/pam.d` directory.

In order to partition access to information and resources on the system, users need to be assigned to appropriate groups granting them any required access. The number and assignments to groups should be decided during the system security planning process, and then configured in the appropriate information repository, whether locally using the configuration files in `/etc`, or on some centralized database. At this time, any default or generic users supplied with the system should be checked, and removed if not required. Other accounts that are required, but are not associated with a user that needs to login, should have login capability disabled, and any associated password or authentication credential removed.

Guides to hardening Unix and Linux systems also often recommend changing the access permissions for critical directories and files, in order to further limit access to them. Programs that set user (setuid) to root or set group (setgid) to a privileged group are key target for attackers. As we discuss in detail in Sections 4.4 and 25.3, such programs execute with superuser rights, or with access to resources belonging to the privileged group, no matter which user executes them. A software vulnerability in such a program can potentially be exploited by an attacker to gain these elevated privileges. This is known as a local exploit. A software vulnerability in a network server could be triggered by a remote attacker. This is known as a remote exploit.

It is widely accepted that the number and size of setuid root programs in particular should be minimized. They cannot be eliminated, as superuser privileges are required to access some resources on the system. The programs that manage user login, and allow network services to bind to privileged ports, are examples. However, other programs, that were once setuid root for programmer convenience, can function as well if made setgid to a suitable privileged group that has the necessary access to some resource. Programs to display system state, or deliver mail, have been modified in this way. System hardening guides may recommend further changes, and indeed the removal of some such programs that are not required on a particular system.

### Remote Access Controls

Given that remote exploits are of concern, it is important to limit access to only those services required. This function may be provided by a perimeter firewall, as we discussed in Chapter 9. However, host-based firewall or network access control mechanisms may provide additional defences. Unix and Linux systems support several alternatives for this.

The TCP Wrappers library and tcpd daemon provide one mechanism that network servers may use. Lightly loaded services may be "wrapped" using `tcpd`, which listens for connection requests on their behalf. It checks that any request is permitted by configured policy before accepting it and invoking the server program to handle it. Requests that are rejected are logged. More complex and heavily loaded servers incorporate this functionality into their own connection management code, using the TCP Wrappers library, and the same policy configuration files. These files are `/etc/hosts.allow` and `/etc/hosts.deny`, which should be set as policy requires.

There are several host firewall programs that may be used. Linux systems primarily use the `iptables` program to configure the `netfilter` kernel module. This provides comprehensive, though complex, stateful packet filtering, monitoring, and modification capabilities. BSD-based systems (including MacOS) now use the pf program with similar capabilities. Most systems provide an administrative utility to generate common configurations and to select which services will be permitted to access the system. These should be used unless there are non-standard requirements, given the skill and knowledge needed to run these programs to edit their configuration files.

### Logging and Log Rotation

Most applications can be configured to log with levels of detail ranging from "debugging" (maximum detail) to "none." Some middle setting is usually the best choice, but you should not assume that the default setting is necessarily appropriate.

In addition, many applications allow you to specify either a dedicated file to write application event data to, or a syslog facility to use when writing log data to `/dev/log` (see Section 25.5). If you wish to handle system logs in a consistent, centralized manner, it is usually preferable for applications to send their log data to `/dev/log`. Note, however, that `logrotate` (also discussed in Section 25.5) can be configured to rotate *any* logs on the system, whether written by `syslogd`, `Syslog-NG`, or individual applications.

### Application Security Using a chroot jail

Some network accessible services do not require access to the full file-system, but rather only need a limited set of data files and directories for their operation. FTP is a common example of such a service. It provides the ability to download files from, and upload files to, a specified directory tree. If such a server were compromised and had access to the entire system, an attacker could potentially access and compromise data elsewhere. Unix and Linux systems provide a mechanism to run such services in a **chroot** jail, which restricts the server's view of the file system to just a specified portion. This is done using the **chroot** system call that confines a process to some subset of the file system by mapping the root of the filesystem "`/`" to some other directory (e.g., `/srv/ftp/public`). To the "chrooted" server, everything in this chroot jail appears to actually be in / (e.g., the "real" directory `/srv/ftp/public/etc/myconfigfile` appears as `/etc/myconfigfile` in the chroot jail). Files in directories outside the chroot jail (e.g., `/srv/www or /etc.`) are not visible or reachable at all.

Chrooting therefore helps contain the effects of a given server being compromised or hijacked. The main disadvantage of this method is added complexity: a number of files (including all executable libraries used by the server), directories, and devices needed must be copied into the chroot jail. Determining just what needs to go into the jail for the server to work properly can be tricky, though detailed procedures for chrooting many different applications are available.

Troubleshooting a chrooted application can also be difficult. Even if an application explicitly supports this feature, it may behave in unexpected ways when run chrooted. Note also that if the chrooted process runs as root, it can "break out" of

the chroot jail with little difficulty. Still, the advantages usually far outweigh the disadvantages of chrooting network services.

### Security Testing

The system hardening guides such as those provided by the "NSA—Security Configuration Guides" include security checklists for a number of Unix and Linux distributions that may be followed.

There are also a number of commercial and open-source tools available to perform system security scanning and vulnerability testing. One of the best known is "Nessus." This was originally an open-source tool, which was commercialized in 2005, though some limited free-use versions are available. "Tripwire" is a well-known file integrity checking tool that maintains a database of cryptographic hashes of monitored files, and scans to detect any changes, whether as a result of malicious attack, or simply accidental or incorrectly managed update. This again was originally an open-source tool, which now has both commercial and free variants available. The "Nmap" network scanner is another well-known and deployed assessment tool that focuses on identifying and profiling hosts on the target network, and the network services they offer.

## 12.7 WINDOWS SECURITY

We now consider some specific issues with the secure installation, configuration, and management of Microsoft Windows systems. These systems have for many years formed a significant portion of all "general purpose" system installations. Hence, they have been specifically targeted by attackers, and consequently security countermeasures are needed to deal with these challenges. The process of providing appropriate levels of security still follows the general outline we describe in this chapter. Beyond the general guidance in this section, we will provide more detailed discussion of Windows security mechanisms in Chapter 26.

Again, there are a large range of resources available to assist **administrators** of these systems, including online resources such as the "Microsoft Security Tools & Checklists," and specific system hardening guides such as those provided by the "NSA—Security Configuration Guides."

### Patch Management

The "Windows Update" service and the "Windows Server Update Services" assist with the regular maintenance of Microsoft software, and should be configured and used. Many other third-party applications also provide automatic update support, and these should be enabled for selected applications.

### Users Administration and Access Controls

Users and groups in Windows systems are defined with a Security ID (SID). This information may be stored and used locally, on a single system, in the Security Account Manager (SAM). It may also be centrally managed for a group of systems belonging to a domain, with the information supplied by a central Active Directory

(AD) system using the LDAP protocol. Most organizations with multiple systems will manage them using domains. These systems can also enforce common policy on users on any system in the domain. We will further explore the Windows security architecture in Section 26.1.

Windows systems implement discretionary access controls to system resources such as files, shared memory, and named pipes. The access control list has a number of entries that may grant or deny access rights to a specific SID, which may be for an individual user or for some group of users. Windows Vista and later systems also include mandatory integrity controls. These label all objects, such as processes and files, and all users, as being of low, medium, high, or system integrity level. Then whenever data is written to an object, the system first ensures that the subject's integrity is equal or higher than the object's level. This implements a form of the Biba Integrity model we will discuss in Section 27.2 that specifically targets the issue of untrusted remote code executing in, for example Windows Internet Explorer, trying to modify local resources.

Windows systems also define privileges, which are system wide and granted to user accounts. Examples of privileges include the ability to backup the computer (which requires overriding the normal access controls to obtain a complete backup), or the ability to change the system time. Some privileges are considered dangerous, as an attacker may use them to damage the system. Hence, they must be granted with care. Others are regarded as benign, and may be granted to many or all user accounts.

As with any system, hardening the system configuration can include further limiting the rights and privileges granted to users and groups on the system. As the access control list gives deny entries greater precedence, you can set an explicit deny permission to prevent unauthorized access to some resource, even if the user is a member of a group that otherwise grants access.

When accessing files on a shared resource, a combination of share and NTFS **permissions** may be used to provide additional security and granularity. For example, you can grant full control to a share, but read-only access to the files within it. If access-based enumeration is enabled on shared resources, it can automatically hide any objects that a user is not permitted to read. This is useful with shared folders containing many users' home directories, for example.

You should also ensure users with administrative rights only use them when required, and otherwise access the system as a normal user. The User Account Control (UAC) provided in Vista and later systems assists with this requirement. These systems also provide Low Privilege Service Accounts that may be used for long-lived service processes, such as file, print, and DNS services that do not require elevated privileges.

### Application and Service Configuration

Unlike Unix and Linux systems, much of the configuration information in Windows systems is centralized in the Registry, which forms a database of keys and values that may be queried and interpreted by applications on these systems.

Changes to these values can be made within specific applications, setting preferences in the application that are then saved in the registry using the appropriate keys and values. This approach hides the detailed representation from the administrator.

Alternatively, the registry keys can be directly modified using the "Registry Editor." This approach is more useful for making bulk changes, such as those recommended in hardening guides. These changes may also be recorded in a central repository, and pushed out whenever a user logs in to a system within a network domain.

### Other Security Controls

Given the predominance of malware that targets Windows systems, it is essential that suitable anti-virus, anti-spyware, personal firewall, and other malware and attack detection and handling software packages are installed and configured on such systems. This is clearly needed for network connected systems, as shown by the high-incidence numbers in reports such as [SYMA16]. However, as the Stuxnet attacks in 2010 show, even isolated systems updated using removable media are vulnerable, and thus must also be protected.

Current generation Windows systems include some basic firewall and malware countermeasure capabilities, which should certainly be used at a minimum. However, many organizations find that these should be augmented with one or more of the many commercial products available. One issue of concern is undesirable interactions between anti-virus and other products from multiple vendors. Care is needed when planning and installing such products to identify possible adverse interactions, and to ensure the set of products in use are compatible with each other.

Windows systems also support a range of cryptographic functions that may be used where desirable. These include support for encrypting files and directories using the Encrypting File System (EFS), and for full-disk encryption with AES using BitLocker.

### Security Testing

The system hardening guides such as those provided by the "NSA—Security Configuration Guides" also include security checklists for various versions of Windows.

There are also a number of commercial and open-source tools available to perform system security scanning and vulnerability testing of Windows systems. The "Microsoft Baseline Security Analyzer" is a simple, free, easy-to-use tool that aims to help small- to medium-sized businesses improve the security of their systems by checking for compliance with Microsoft's security recommendations. Larger organizations are likely better served using one of the larger, centralized, commercial security analysis suites available.

## 12.8 VIRTUALIZATION SECURITY

**Virtualization** refers to a technology that provides an abstraction of the computing resources used by some software, which thus runs in a simulated environment called a virtual machine (VM). There are many types of virtualization; however, in this section we are most interested in full virtualization. This allows multiple full operating system instances to execute on virtual hardware, supported by a hypervisor that manages

access to the actual physical hardware resources. Benefits arising from using virtualization include better efficiency in the use of the physical system resources than is typically seen using a single operating system instance. This is particularly evident in the provision of virtualized server systems. Virtualization can also provide support for multiple distinct operating systems and associated applications on the one physical system. This is more commonly seen on client systems.

There are a number of additional security concerns raised in virtualized systems, as a consequence both of the multiple operating systems executing side by side and of the presence of the virtualized environment and hypervisor as a layer below the operating system kernels and the security services they provide. [CLEE09] presents a survey of some of the security issues arising from such a use of virtualization, a number of which we will discuss further.

### Virtualization Alternatives

The hypervisor is software that sits between the hardware and the VMs and acts as a resource broker. Simply put, it allows multiple VMs to safely coexist on a single physical server host and share that host's resources. The virtualizing software provides abstraction of all physical resources (such as processor, memory, network, and storage) and thus enables multiple computing stacks, called virtual machines, to be run on a single physical host.

Each VM includes an OS, called the guest OS. This OS may be the same as the host OS, if present, or a different one. For example, a guest Windows OS could be run in a VM on top of a Linux host OS. The guest OS, in turn, supports a set of standard library functions and other binary files and applications. From the point of view of the applications and the user, this stack appears as an actual machine, with hardware and an OS; thus the term *virtual machine* is appropriate. In other words, it is the hardware that is being virtualized.

The principal functions performed by a hypervisor are the following:

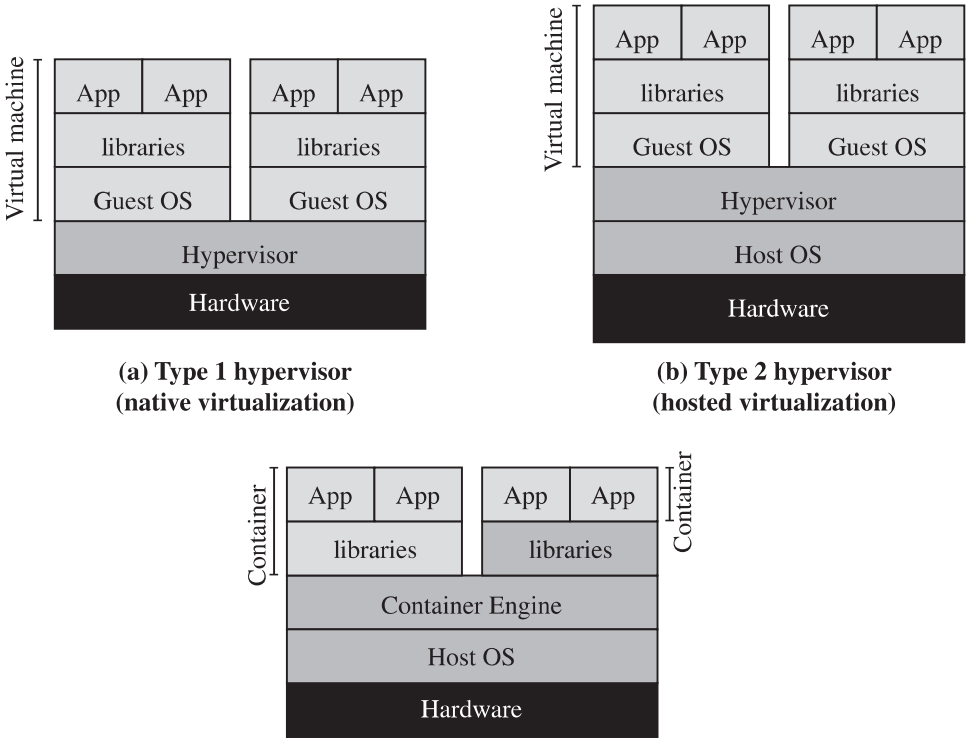
- **Execution management of VMs**: Includes scheduling VMs for execution, virtual memory management to ensure VM isolation from other VMs, and context switching between various processor states. Also includes isolation of VMs to prevent conflicts in resource usage and emulation of timer and interrupt mechanisms.
- **Devices emulation and access control**: Emulating all network and storage (block) devices that different native drivers in VMs are expecting, and mediating access to physical devices by different VMs.
- **Execution of privileged operations by hypervisor for guest VMs**: Certain operations invoked by guest OSs, instead of being executed directly by the host hardware, may have to be executed on its behalf by the hypervisor, because of their privileged nature.
- **Management of VMs (also called VM lifecycle management)**: Configuring guest VMs and controlling VM states (e.g., Start, Pause, Stop).
- **Administration of hypervisor platform and hypervisor software**: Involves setting of parameters for user interactions with the hypervisor host as well as hypervisor software.

*TYPE 1 HYPERVISOR* There are two types of hypervisors, distinguished by whether there is an OS between the hypervisor and the host. A **type 1 hypervisor** (see Figure 12.2a) is loaded as a software layer directly onto a physical server, much like an OS is loaded; this is referred to as **native virtualization**. The type 1 hypervisor can directly control the physical resources of the host. Once it is installed and configured, the server is then capable of supporting virtual machines as guests. In mature environments, where virtualization hosts are clustered together for increased availability and load balancing, a hypervisor can be staged on a new host. Then, that new host is joined to an existing cluster, and VMs can be moved to the new host without any interruption of service.

*TYPE 2 HYPERVISOR* A **type 2 hypervisor** exploits the resources and functions of a host OS and runs as a software module on top of the OS (see Figure 12.2b); this is referred to as **hosted virtualization**. It relies on the OS to handle all of the hardware interactions on the hypervisor's behalf.

Key differences between the two hypervisor types are as follows:

- Typically, type 1 hypervisors perform better than type 2 hypervisors. Because a Type 1 hypervisor doesn't compete for resources with an OS, there are more resources available on the host, and by extension, more virtual machines can be hosted on a virtualization server using a Type 1 hypervisor.
- Type 1 hypervisors are also considered to be more secure than the Type 2 hypervisors. Virtual machines on a Type 1 hypervisor make resource requests that are handled external to that guest, and they cannot affect other VMs or the




**(a) Type 1 hypervisor (native virtualization)**

**(b) Type 2 hypervisor (hosted virtualization)**

**(c) Container (application virtualization)**

**Figure 12.2 Comparison of Virtual Machines and Containers**

hypervisor they are supported by. This is not necessarily true for VMs on a Type 2 hypervisor and a malicious guest could potentially affect more than itself.

- Type 2 hypervisors allow a user to take advantage of virtualization without needing to dedicate a server to only that function. Developers who need to run multiple environments as part of their process, in addition to taking advantage of the personal productive workspace that a PC OS provides, can do both with a type 2 hypervisor installed as an application on their LINUX, MacOSX, or Windows desktop. The virtual machines that are created and used can be migrated or copied from one hypervisor environment to another, reducing deployment time and increasing the accuracy of what is deployed, and reducing the time to market a project.

**Native virtualization** systems are typically seen in servers, with the goal of improving the execution efficiency of the hardware. They are arguably also more secure, as they have fewer additional layers than the alternative hosted approach. **Hosted virtualization** systems are more common in clients, where they run alongside other applications on the host OS, and are used to support applications for alternate operating system versions or types.

In virtualized systems, the available hardware resources must be appropriately shared among the various guest OS's. These include CPU, memory, disk, network, and other attached devices. CPU and memory are generally partitioned between these, and scheduled as required. Disk storage may be partitioned, with each guest having exclusive use of some disk resources. Alternatively, a "virtual disk" may be created for each guest, which appears to it as a physical disk with a full file-system, but is viewed externally as a single "disk image" file on the underlying file-system. Attached devices such as optical disks, or USB devices are generally allocated to a single guest OS at a time.

Several alternatives exist for providing network access. The guest OS may have direct access to distinct network interface cards on the system; the hypervisor may mediate access to shared interfaces; or the hypervisor may implement virtual network interface cards for each guest, bridging or routing traffic between guests as required. This last approach uses one or more virtual network switches, which are implemented in the hypervisor kernel, and is quite common. It is arguably the most efficient approach, since traffic between guests does not need to be relayed via external network links. It does have security consequences in that this traffic is not subject to monitoring by probes attached to physical networks, such as we discussed in Chapter 9.

When a number virtualized systems and hypervisors are grouped together in a data center, or even between data centers, the various systems need to connect to appropriate network segments, with suitable routing and firewalls connecting them together, and to the Internet. The cloud computing solutions we will discuss in Chapter 13 use this structure, as do computing solutions for some large organizations. The network connections can be made with physical, external, links, using IDS and firewalls to link them together as we discussed in Chapters 8 and 9. However this approach limits the flexibility of the virtualized solution, as virtual machines can only be migrated to other hosts with the required physical network connections already in place. VLANs can provide more flexibility in the network architecture, though are still limited by the physical network connections and VLAN configuration.

Greater flexibility still is provided by **software defined networks** (SDNs), which enable network segments to logically span multiple servers within and between data centers, while using the same underlying physical network. There are several possible approaches to providing SDNs, including the use of **overlay networks**. These abstract all layer 2 and 3 addresses from the underlying physical network into whatever logical network structure is required. And this structure can be easily changed and extended as needed. The IETF standard DOVE (Distributed Overlay Virtual Network), which uses VXLAN (Virtual Extended Local Area Network) can be used to implement such an overlay network. With this flexible structure, it is possible to locate virtual servers, virtual IDS, and virtual firewalls anywhere within the network as required. We further discuss the use of secure virtual networks and firewalls later in this section.

*CONTAINERS* A relatively recent approach to virtualization, known as **container virtualization** or **application virtualization**, is worth noting (see Figure 12.2c). In this approach, software, known as a **virtualization container**, runs on top of the host OS kernel and provides an isolated execution environment for applications. Unlike hypervisor-based VMs, containers do not aim to emulate physical servers. Instead, all containerized applications on a host share a common OS kernel. This eliminates the resources needed to run a separate OS for each application and can greatly reduce overhead.

For containers, only a small container engine is required as support for the containers. The container engine sets up each container as an isolated instance by requesting dedicated resources from the OS for each container. Each container app then directly uses the resources of the host OS. VM virtualization functions at the border of hardware and OS. It's able to provide strong performance isolation and security guarantees with the narrowed interface between VMs and hypervisors. Containerization, which sits in between the OS and applications, incurs lower overhead, but potentially introduces greater security vulnerabilities.

## Virtualization Security Issues

[CLEE09] and NIST SP 800-125 (*Guide to Security for Full Virtualization Technologies*, January 2011) both detail a number of security concerns that result from the use of virtualized systems, including:

- Guest OS isolation, ensuring that programs executing within a guest OS may only access and use the resources allocated to it, and not covertly interact with programs or data either in other guest OSs or in the hypervisor.
- Guest OS monitoring by the hypervisor, which has privileged access to the programs and data in each guest OS, and must be trusted as secure from subversion and compromised use of this access.
- Virtualized environment security, particularly image and snapshot management, which attackers may attempt to view or modify.

These security concerns may be regarded as an extension of the concerns we have already discussed with securing operating systems and applications. If a particular operating system and application configuration is vulnerable when running directly on hardware in some context, it will most likely also be vulnerable when running

in a virtualized environment. And should that system actually be compromised, it would be at least as capable of attacking other nearby systems, whether they are also executing directly on hardware or running as other guests in a virtualized environment. The use of a virtualized environment may improve security by further isolating network traffic between guests than would be the case when such systems run natively, however this traffic is not visible to external IDS or firewall systems, and may require the use of virtual firewalls to manage. Further the ability of the hypervisor to transparently monitor activity on all guests OS may be used as a form of virtual firewall or IDS to assist in securing these systems. However, the presence of the virtualized environment and the hypervisor may reduce security if vulnerabilities exist within it which attackers may exploit. Such vulnerabilities could allow programs executing in a guest to covertly access the hypervisor, and hence other guest OS resources. This is known as VM escape, and is of concern, as we discussed in Section 6.8. Virtualized systems also often provide support for suspending an executing guest OS in a snapshot, saving that image, and then restarting execution at a later time, possibly even on another system. If an attacker can view or modify this image, they can compromise the security of the data and programs contained within it. The use of infrastructure with many virtualized systems within and between data centers, linked using software-defined networks, raise further security concerns.

Thus, the use of virtualization adds additional layers of concern, as we have previously noted. Securing virtualized systems means extending the security process to secure and harden these additional layers. In addition to securing each guest operating system and applications, the virtualized environment and the hypervisor must also be secured.

### Securing Virtualization Systems

NIST SP 800-125 provides guidance for providing appropriate security in virtualized systems, and states that organizations using virtualization should:

- Carefully plan the security of the virtualized system.
- Secure all elements of a full virtualization solution, including the hypervisor, guest OSs, and virtualized infrastructure, and maintain their security.
- Ensure that the hypervisor is properly secured.
- Restrict and protect administrator access to the virtualization solution.

This is clearly seen as an extension of the process of securing systems that we presented earlier in this chapter.

***Hypervisor Security*** The hypervisor should be secured using a process similar to that with securing an operating system. That is, it should be installed in an isolated environment, from known clean media, and updated to the latest patch level in order to minimize the number of vulnerabilities that may be present. It should then be configured so that it is updated automatically, any unused services are disabled or removed, unused hardware is disconnected, appropriate introspection capabilities are used with the guest OSs, and the hypervisor is monitored for any signs of compromise.

Access to the hypervisor should be limited to authorized administrators only, since these users would be capable of accessing and monitoring activity in any of the guest OSs. The hypervisor may support both local and remote administration. This must be configured appropriately, with suitable authentication and encryption mechanisms used, particularly when using remote administration. Remote administration access should also be considered and secured in the design of any network firewall and IDS capability in use. Ideally such administration traffic should use a separate network, with very limited, if any, access provided from outside the organization.

### Virtualized Infrastructure Security

The wider virtualization infrastructure must be carefully managed and configured. Virtualized system hypervisors manage access to hardware resources such as disk storage and network interfaces. This access must be limited to just the appropriate guest OSs that use any resource, and network connections suitably arranged. Access to VM images and snapshots must also be carefully controlled, since these are another potential point of attack.

When multiple virtualized systems are used, NIST SP 800-125B (*Secure Virtual Network Configuration for Virtual Machine (VM) Protection*, March 2016) notes three distinct categories of network traffic:

- **Management traffic**: used for hypervisor administration and configuration of the virtualized infrastructure.
- **Infrastructure traffic**: such as migration of VM images, or connections to network storage technologies.
- **Application traffic**: between applications running VMs and to external networks. This traffic may be further separated into a number of segments, isolating traffic from applications with different sensitivity levels, or from different organizations or departments.

Traffic in each of these should be suitably isolated and protected. This requires the use of a number of network segments, connected as needed by appropriate firewall systems. These may variously use a combination of distinct physical network connections, VLANs, or software defined networks to provide a suitable network structure. For example, in larger installations, management and infrastructure traffic may use relatively static physical network connections, while the application traffic would use more flexible VLANs or software defined networks layered over a separate base physical network structure.

### Virtual Firewall

As we mentioned in Section 9.4, a **Virtual Firewall** provides firewall capabilities for the network traffic flowing between systems hosted in a virtualized or cloud environment that does not require this traffic to be routed out to a physically separate network supporting traditional firewall services. These capabilities may be provided by a combination of:

- **VM Bastion Host**: Where a separate VM is used as a bastion host supporting the same firewall systems and services that could be configured to run on a

physically separate bastion, including possibly IDS and IPS services. The network connections used by other VMs are configured to connect them to suitable sub-networks. These are connected to distinct virtual network interfaces on the VM Bastion Host, which can monitor and route traffic between them in the same manner, and with the same configuration possibilities, as on a physically separate bastion host. Such systems may be provided as a virtual UTM installed into a suitably hardened VM that can be easily loaded, configured, and run as needed. A disadvantage of this approach is that these virtual bastions compete for the same hypervisor host resources as other VMs on that system.

- **VM Host-Based Firewall**: Where host-based firewall capabilities provided by the Guest OS running on the VM are configured to secure that host in the same manner as used in physically separate systems.
- **Hypervisor Firewall**: Where firewall capabilities are provided directly by the hypervisor. These capabilities range from stateless or stateful packet inspection in the virtual network switches that forward network traffic between VMs, to a full hypervisor firewall capable of monitoring all activity within its VMs. This latter variant provides capabilities of both host-based and bastion host firewalls, but from a location outside the traditional host and network structure. It can be more secure than the other alternatives, as it is not part of the virtualized network, nor visible as a separate VM. It may also be more efficient than the alternatives, since the resource monitoring and filtering occur within the hypervisor kernel running directly on the hardware. However, it requires a hypervisor that supports these features, which also adds to its complexity.

When used in large-scale virtualized environments, with many virtualized systems linked with VLANs or software defined networks across one or more data centers, virtual firewall bastions can be provisioned and located as needed where suitable resources are available. This provides a greater level of flexibility and scalability than many traditional structures can support. However, there may still be a need for some physical firewall systems, especially to support very high traffic volumes either between virtual servers or on their connection to the wider Internet.

***Hosted Virtualization Security*** Hosted virtualized systems, as typically used on client systems, pose some additional security concerns. These result from the presence of the host OS under, and other host applications beside, the hypervisor and its guest OSs. Hence there are yet more layers to secure. Further, the users of such systems often have full access to configure the hypervisor, and to any VM images and snapshots. In this case, the use of virtualization is more to provide additional features, and to support multiple operating systems and applications, than to isolate these systems and data from each other, and from the users of these systems.

It is possible to design a host system and virtualization solution that is more protected from access and modification by the users. This approach may be used to support well-secured guest OS images used to provide access to enterprise networks and data, and to support central administration and update of these images. However, there will remain security concerns from possible compromise of the underlying host OS, unless it is adequately secured and managed.

## 12.9 KEY TERMS, REVIEW QUESTIONS, AND PROBLEMS

### Key Terms

| | | |
|---|---|---|
| access controls | guest OS | patching |
| administrators | hardening | permissions |
| application virtualization | hosted virtualization | software defined network |
| archive | hypervisor | type 1 hypervisor |
| backup | logging | type 2 hypervisor |
| chroot | native virtualization | virtualization |
| container virtualization | overlay network | |
| full virtualization | patches | |

### Review Questions

**12.1** What are the basic steps needed in the process of securing a system?
**12.2** What is "hardening"?
**12.3** What are the basic steps needed to secure the base operating system?
**12.4** Why is keeping all software as up to date as possible so important?
**12.5** What are the pros and cons of automated patching?
**12.6** Why is it better to not install software applications from unknown sources at all, instead of installing them, perhaps testing them, and then removing or disabling them?
**12.7** What types of additional security controls may be used to secure the base operating system?
**12.8** What additional steps are used to secure key applications?
**12.9** What steps are used to maintain system security?
**12.10** Why is effective logging considered a cornerstone of sound security practice?
**12.11** What is the difference between a data backup and data archiving?
**12.12** Where is user account and group information stored in Unix systems?
**12.13** How does the Windows operating system provide patch management?
**12.14** What effect do set user and set group permissions have when executing files on Unix and Linux systems?
**12.15** What is the main host firewall program used on Linux systems?
**12.16** What is meant by a tripwire?
**12.17** How is a chroot jail used to improve application security on Unix and Linux systems?
**12.18** Where are two places user and group information may be stored on Windows systems?
**12.19** What are the major differences between the implementations of the discretionary access control models on Unix and Linux systems and those on Windows systems?
**12.20** What are mandatory integrity controls used for in Windows systems?
**12.21** On Windows, which privilege overrides all ACL checks, and why?
**12.22** Where is application and service configuration information stored on Windows systems?
**12.23** What is a hypervisor?
**12.24** State different types of full virtualization with their security requirements.

12.25 What are the main security concerns with a hypervisor?
12.26 What is VM escape and what are its implications?

## Problems

12.1 Describe the main reason for not eliminating the setuid root programs completely from the operating systems.
12.2 Set user (setuid) and set group (setgid) programs and scripts are a powerful mechanism provided by Unix to support "controlled invocation" to manage access to sensitive resources. However, precisely because of this it is a potential security hole, and bugs in such programs have led to many compromises on Unix systems. Detail a command you could use to locate all set user or group scripts and programs on a Unix system, and how you might use this information.
12.3 How can we use the TCP Wrappers and tcpd daemon to achieve secure remote control access? What if the network servers are heavily loaded?
12.4 Employee "david" owns a directory, "exams," containing a text file called "papers.txt" that he shares with users belonging to the group "examiners." Those users can not only read and change this file, but also delete it. They can add other files to the directory. Others may neither read, write, nor execute anything in "examiners." What would appropriate ownerships and permissions for both the directory "examiners" and the file "papers.txt" look like? (Write your answers in the form of "long listing" output.)
12.5 Suppose you operate an Apache-based Linux Web server that hosts your company's e-commerce site. Suppose further there is a worm called "WorminatorX," which exploits a (fictional) buffer overflow bug in the Apache Web server package that can result in a remote root compromise. Construct a simple threat model that describes the risk this represents: attacker(s), attack-vector, vulnerability, assets, likelihood of occurrence, likely impact, and plausible mitigations.
12.6 Why is it important to secure the boot process? Is it required to limit which media the system must boot from?
12.7 Consider an automated audit log analysis tool (e.g., swatch). Can you propose some rules which could be used to distinguish "suspicious activities" from normal user behavior on a system for some organization?
12.8 Assume a hosted virtualization system in which a hypervisor executes and manages a total of six guest operating systems. Suppose an external hard disk is attached to the system and three guest operating systems need to access it for retrieving data. Will the attached hard disk be allocated to all the three guest operating systems at the same time? Moreover, how will the hypervisor provide network access to the guest operating systems if the total number of network interface cards attached to the system is not enough?
12.9 Some have argued that Unix/Linux systems reuse a small number of security features in many contexts across the system, while Windows systems provide a much larger number of more specifically targeted security features used in the appropriate contexts. This may be seen as a trade-off between simplicity and lack of flexibility in the Unix/Linux approach, against a better targeted but more complex and harder to correctly configure approach in Windows. Discuss this trade-off as it impacts on the security of these respective systems, and the load placed on administrators in managing their security.
12.10 It is recommended that while using a hypervisor, the access to the hypervisor should be limited to authorized administrators only. Why?

CHAPTER 13

# Cloud and IoT Security

**Learning Objectives**

After studying this chapter, you should be able to:

- Present an overview of cloud computing concepts.
- List and define the principal cloud services.
- List and define the cloud deployment models.
- Explain the NIST cloud computing reference architecture.
- Describe Cloud Security as a Service.
- Understand the OpenStack security module for cloud security.
- Explain the scope of the Internet of things.
- List and discuss the five principal components of IoT-enabled things.
- Understand the relationship between cloud computing and IoT.
- Define the patching vulnerability.
- Explain the IoT Security Framework.
- Understand the MiniSec security feature for wireless sensor networks.

The two most significant developments in computing in recent years are cloud computing and the Internet of Things (IoT). In both cases, security measures tailored to the specific requirements of these environments are evolving. This chapter begins with an overview of the concepts of cloud computing, followed by a discussion of cloud security. Then the chapter examines the concepts of IoT and closes with a discussion of IoT security.

For further detail on the material on cloud computing and IoT in Sections 13.1 and 13.4, see [STAL16a].

## 13.1 CLOUD COMPUTING

There is an increasingly prominent trend in many organizations to move a substantial portion or even all information technology (IT) operations to an Internet-connected infrastructure known as enterprise cloud computing. The use of cloud computing raises a number of security issues, particularly in the area of database security. This section provides an overview of cloud computing. Section 13.2 discussed cloud computing security.

### Cloud Computing Elements

NIST defines cloud computing, in NIST SP 800-145 (*The NIST Definition of Cloud Computing*, September 2011) as follows:

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

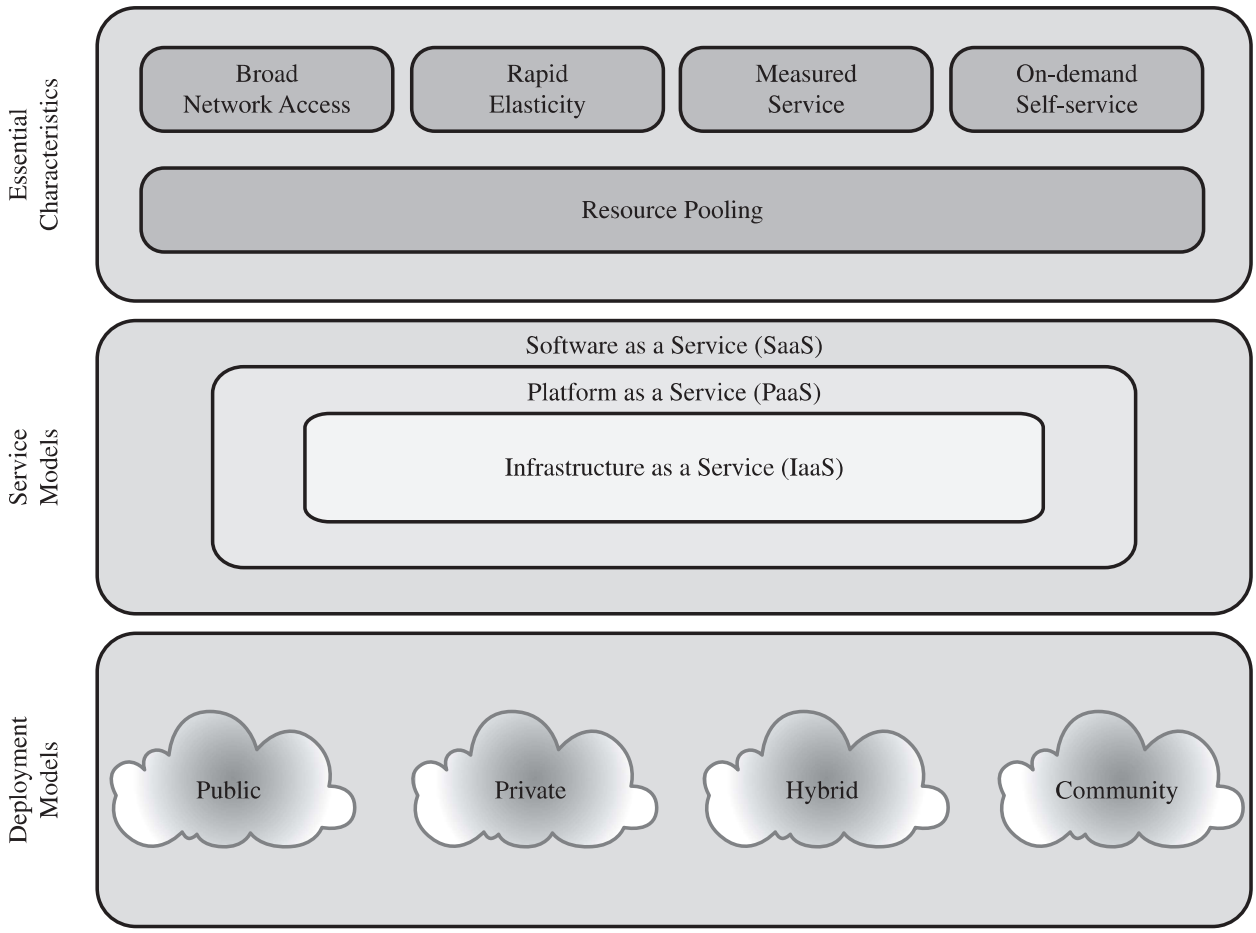



**Figure 13.1** **Cloud Computing Elements**

The definition refers to various models and characteristics, whose relationship is illustrated in Figure 13.1. The essential characteristics of cloud computing includes the following:

- **Broad network access:** Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, laptops, and tablets) as well as other traditional or cloud-based software services.
- **Rapid elasticity:** Cloud computing gives you the ability to expand and reduce resources according to your specific service requirement. For example, you may need a large number of server resources for the duration of a specific task. You can then release these resources upon completion of the task.
- **Measured service:** Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.
- **On-demand self-service:** A cloud service consumer (CSC) can unilaterally provision computing capabilities, such as server time and network storage, as needed, automatically, without requiring human interaction with each service

provider. Because the service is on demand, the resources are not permanent parts of the consumer's IT infrastructure.

- **Resource pooling:** The provider's computing resources are pooled to serve multiple CSCs using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a degree of location independence in that the CSC generally has no control or knowledge of the exact location of the provided resources, but may be able to specify location at a higher level of abstraction (e.g., country, state, or data center). Examples of resources include storage, processing, memory, network bandwidth, and virtual machines (VMs). Even private clouds tend to pool resources between different parts of the same organization.

### Cloud Service Models

NIST SP 800-145 defines three **service models**, which can be viewed as nested service alternatives: Software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS).

***Software as a Service*** SaaS provides service to customers in the form of software, specifically application software, running on and accessible in the cloud. SaaS follows the familiar model of Web services, in this case applied to cloud resources. SaaS enables the customer to use the cloud provider's applications running on the provider's cloud infrastructure. The applications are accessible from various client devices through a simple interface such as a Web browser. Instead of obtaining desktop and server licenses for software products it uses, an enterprise obtains the same functions from the cloud service. The use of SaaS avoids the complexity of software installation, maintenance, upgrades, and patches. Examples of services at this level are Google Gmail, Microsoft 365, Salesforce, Citrix GoToMeeting, and Cisco WebEx.

Common subscribers to SaaS are organizations that want to provide their employees with access to typical office productivity software, such as document management and e-mail. Individuals also commonly use the SaaS model to acquire cloud resources. Typically, subscribers use specific applications on demand. The cloud provider also usually offers data-related features such as automatic backup and data sharing between subscribers.

***Platform as a Service*** A PaaS cloud provides service to customers in the form of a platform on which the customer's applications can run. PaaS enables the customer to deploy onto the cloud infrastructure customer-created or -acquired applications. A PaaS cloud provides useful software building blocks, plus a number of development tools, such as programming language tools, run-time environments, and other tools that assist in deploying new applications. In effect, PaaS is an operating system in the cloud. PaaS is useful for an organization that wants to develop new or tailored applications while paying for the needed computing resources only as needed and only for as long as needed. AppEngine, Engine Yard, Heroku, Microsoft Azure, Force.com, and Apache Stratos are examples of PaaS.

***Infrastructure as a Service*** With IaaS, the customer has access to the resources of the underlying cloud infrastructure. The cloud service user does not manage or control the resources of the underlying cloud infrastructure but has control over

operating systems, deployed applications, and possibly limited control of select networking components (e.g., host firewalls). IaaS provides VMs and other virtualized hardware and operating systems. IaaS offers the customer processing, storage, networks, and other fundamental computing resources so that the customer is able to deploy and run arbitrary software, which can include operating systems and applications. IaaS enables customers to combine basic computing services, such as number crunching and data storage, to build highly adaptable computer systems.

Typically, customers are able to self-provision this infrastructure, using a Web-based graphical user interface that serves as an IT operations management console for the overall environment. API access to the infrastructure may also be offered as an option. Examples of IaaS are Amazon Elastic Compute Cloud (Amazon EC2), Microsoft Windows Azure, Google Compute Engine (GCE), and Rackspace. Figure 13.2 compares the functions implemented by the cloud service provider for the three service models.

## Cloud Deployment Models

There is an increasingly prominent trend in many organizations to move a substantial portion or even all IT operations to enterprise cloud computing. The organization is faced with a range of choices as to cloud ownership and management. In this subsection, we look at the four most prominent deployment models for cloud computing.

***Public Cloud*** A public cloud infrastructure is made available to the general public or a large industry group and is owned by an organization selling cloud services.

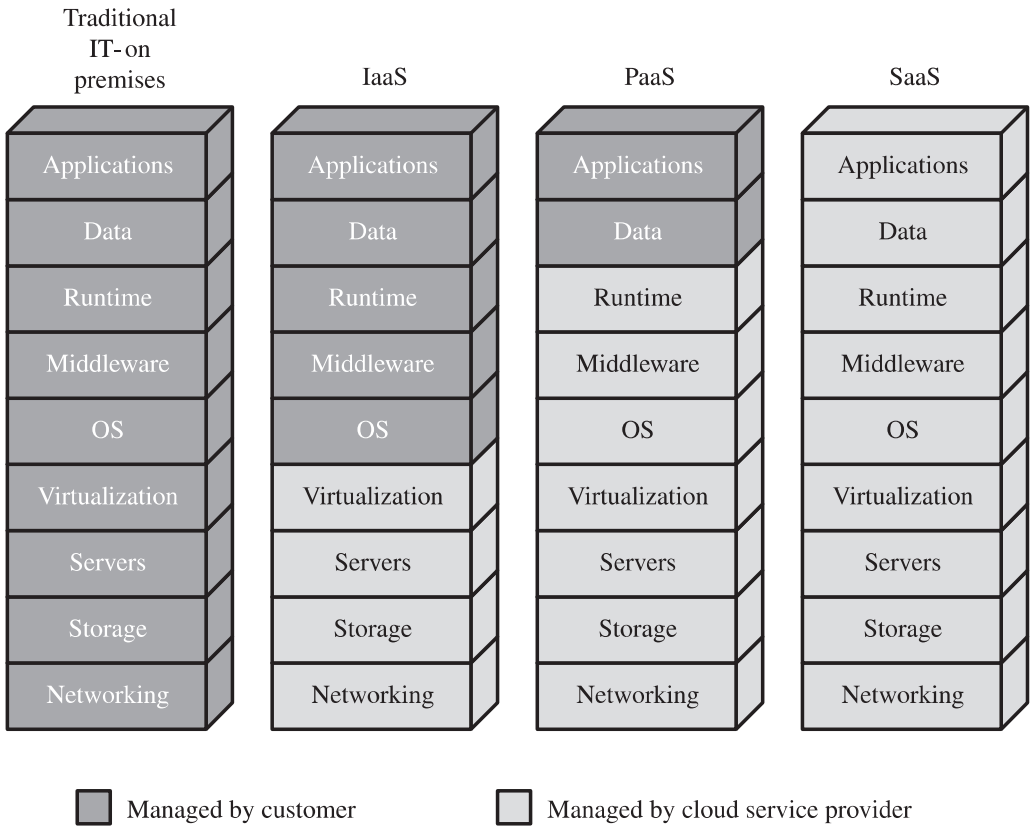



**Figure 13.2 Separation of Responsibilities in Cloud Service Models**

The cloud provider is responsible both for the cloud infrastructure and for the control of data and operations within the cloud. A public cloud may be owned, managed, and operated by a business, academic, government organization, or some combination of them. It exists on the premises of the cloud service provider.

In a public cloud model, all major components are outside the enterprise firewall, located in a multitenant infrastructure. Applications and storage are made available over the Internet via secure IP, and can be free or offered at a pay-per-usage fee. This type of cloud supplies easy-to-use consumer-type services, such as Amazon and Google on-demand Web applications or capacity, Yahoo mail, and Facebook or LinkedIn social media providing free storage for photographs. While public clouds are inexpensive and scale to meet needs, they typically provide no or lower SLAs, and may not offer the guarantees against data loss or corruption found with private or hybrid cloud offerings. The public cloud is appropriate for CSCs and entities not requiring the same levels of service that are expected within the firewall. In addition, the public IaaS clouds do not necessarily provide for restrictions and compliance with privacy laws, which remain the responsibility of the subscriber or corporate end user. In many public clouds, the focus is on the CSC and small and medium-sized businesses where pay-per-use pricing is available, often equating to pennies per gigabyte. Examples of services here might be photo and music sharing, laptop backup or file sharing.

The major advantage of the public cloud is cost. A subscribing organization only pays for the services and resources it needs and can adjust these as needed. Further, the subscriber has greatly reduced management overhead. The principal concern is security. However, there are a number of public cloud providers that have demonstrated strong security controls and, in fact, such providers may have more resources and expertise to devote to security that would be available in a private cloud.

***Private Cloud*** A private cloud is implemented within the internal IT environment of the organization. The organization may choose to manage the cloud in house, or contract the management function to a third party. Additionally, the cloud servers and storage devices may exist on premise, off premise or both.

Private clouds can deliver IaaS internally to employees or business units through an intranet or the Internet via a virtual private network (VPN), as well as software (applications) or storage as services to its branch offices. In both cases, private clouds are a way to leverage existing infrastructure, and deliver and chargeback for bundled or complete services from the privacy of the organization's network. Examples of services delivered through the private cloud include database on demand, e-mail on demand, and storage on demand.

A key motivation for opting for a private cloud is security. A private cloud infrastructure offers tighter controls over the geographic location of data storage and other aspects of security. Other benefits include easy resource sharing and rapid deployment to organizational entities.

***Community Cloud*** A community cloud shares the characteristics of private and public clouds. Like a private cloud, a community cloud has restricted access. Like a public cloud, the cloud resources are shared among a number of independent organizations. The organizations that share the community cloud have similar requirements and, typically, a need to exchange data with each other. One example of an industry that is employing the community cloud concept is the health care industry. A community

**Table 13.1 Comparison of Cloud Deployment Models**

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

cloud can be implemented to comply with government privacy and other regulations. The community participants can exchange data in a controlled fashion.

The cloud infrastructure may be managed by the participating organizations or a third party and may exist on premise or off premise. In this deployment model, the costs are spread over fewer users than a public cloud (but more than a private cloud), so only some of the cost savings potential of cloud computing are realized.

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

A hybrid public/private cloud solution can be particularly attractive for smaller businesses. Many applications for which security concerns are less can be offloaded at considerable cost savings without committing the organization to moving more sensitive data and applications to the public cloud. Table 13.1 lists some of the relative strengths and weaknesses of the four cloud deployment models.

## Cloud Computing Reference Architecture

NIST SP 500–292 (*NIST Cloud Computing Reference Architecture*, September 2011) establishes reference architecture, described as follows:

> The NIST cloud computing reference architecture focuses on the requirements of "what" cloud services provide, not a "how to" design solution and implementation. The reference architecture is intended to facilitate the understanding of the operational intricacies in cloud computing. It does not represent the system architecture of a specific cloud computing system; instead it is a tool for describing, discussing, and developing a system-specific architecture using a common framework of reference.

NIST developed the reference architecture with the following objectives in mind:

- To illustrate and understand the various cloud services in the context of an overall cloud computing conceptual model.

- To provide a technical reference for CSCs to understand, discuss, categorize, and compare cloud services.
- To facilitate the analysis of candidate standards for security, interoperability, and portability and reference implementations.

The reference architecture, depicted in Figure 13.3, defines five major actors in terms of the roles and responsibilities:

- **Cloud service consumer (CSC):** A person or organization that maintains a business relationship with, and uses service from, cloud providers.
- **Cloud service provider (CSP):** A person, organization, or entity responsible for making a service available to interested parties.
- **Cloud auditor:** A party that can conduct independent assessment of cloud services, information system operations, performance, and security of the cloud implementation.
- **Cloud broker:** An entity that manages the use, performance and delivery of cloud services, and negotiates relationships between CSPs and cloud consumers.
- **Cloud carrier:** An intermediary that provides connectivity and transport of cloud services from CSPs to cloud consumers.

The roles of the cloud consumer and provider have already been discussed. To summarize, a **cloud service provider** can provide one or more of the cloud services to meet IT and business requirements of **cloud service consumers**. For each of the three service models (SaaS, PaaS, and IaaS), the CSP provides the storage and processing facilities needed to support that service model, together with a cloud interface for cloud service consumers. For SaaS, the CSP deploys, configures, maintains, and updates the operation of the software applications on a cloud infrastructure so that

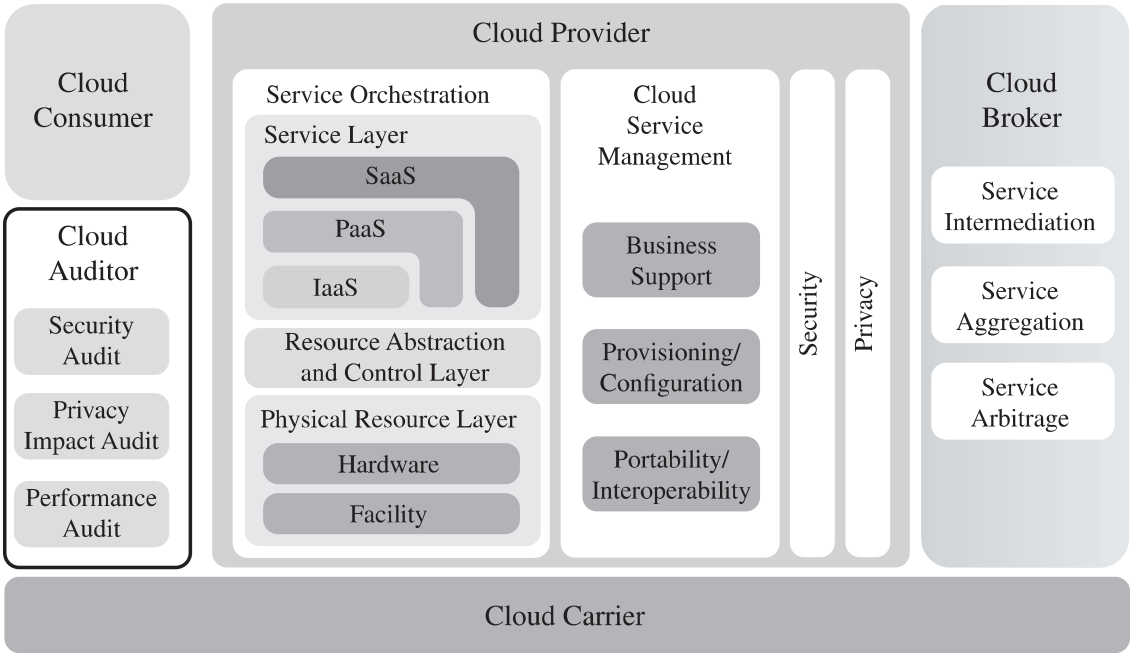



**Figure 13.3 NIST Cloud Computing Reference Architecture**

the services are provisioned at the expected service levels to cloud consumers. The consumers of SaaS can be organizations that provide their members with access to software applications, end users who directly use software applications, or software application administrators who configure applications for end users.

For PaaS, the CSP manages the computing infrastructure for the platform and runs the cloud software that provides the components of the platform, such as runtime software execution stack, databases, and other middleware components. Cloud consumers of PaaS can employ the tools and execution resources provided by CSPs to develop, test, deploy, and manage the applications hosted in a cloud environment.

For IaaS, the CSP acquires the physical computing resources underlying the service, including the servers, networks, storage, and hosting infrastructure. The IaaS CSC in turn uses these computing resources, such as a virtual computer, for their fundamental computing needs.

The **cloud carrier** is a networking facility that provides connectivity and transport of cloud services between cloud consumers and CSPs. Typically, a CSP will set up service level agreements (SLAs) with a cloud carrier to provide services consistent with the level of SLAs offered to cloud consumers, and may require the cloud carrier to provide dedicated and secure connections between cloud consumers and CSPs.

A **cloud broker** is useful when cloud services are too complex for a cloud consumer to easily manage. A cloud broker can offer three areas of support are as follows:

- **Service intermediation:** These are value-added services, such as identity management, performance reporting, and enhanced security.
- **Service aggregation:** The broker combines multiple cloud services to meet consumer needs not specifically addressed by a single CSP, or to optimize performance or minimize cost.
- **Service arbitrage:** This is similar to service aggregation except that the services being aggregated are not fixed. Service arbitrage means a broker has the flexibility to choose services from multiple agencies. The cloud broker, for example, can use a credit-scoring service to measure and select an agency with the best score.

A **cloud auditor** can evaluate the services provided by a CSP in terms of security controls, privacy impact, performance, and so on. The auditor is an independent entity that can assure that the CSP conforms to a set of standards.

Figure 13.4 illustrates the interactions between the actors. A cloud consumer may request cloud services from a cloud provider directly or via a cloud broker. A cloud auditor conducts independent audits and may contact the others to collect necessary information. This figure shows that cloud networking issues involve three separate types of networks. For a cloud producer, the network architecture is that of a typical large data center, which consists of racks of high-performance servers and storage devices, interconnected with high-speed top-of-rack Ethernet switches. The concerns in this context focus on VM placement and movement, load balancing, and availability issues. The enterprise network is likely to have a quite different architecture, typically including a number of LANs, servers, workstations, PCs, and mobile devices, with a broad range of network performance, security, and management issues.

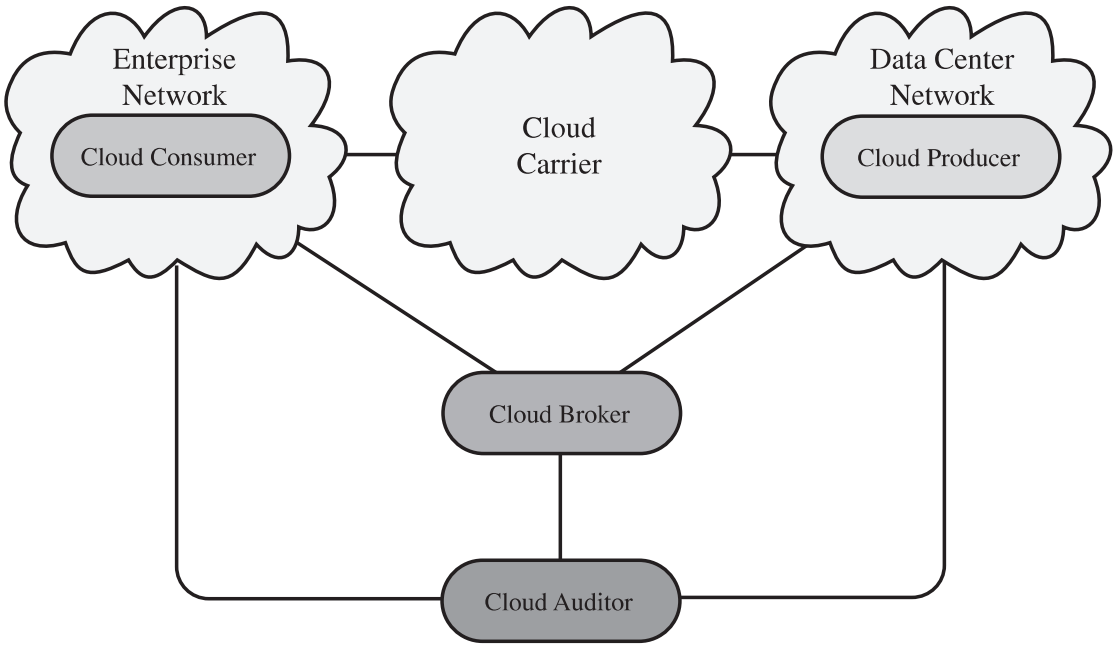



Figure 13.4 **Interactions between Actors in Cloud Computing**

The concern of both producer and consumer with respect to the cloud carrier, which is shared with many users, is the ability to create virtual networks with appropriate SLA and security guarantees.

## 13.2 CLOUD SECURITY CONCEPTS

There are numerous aspects to cloud security and numerous approaches to providing cloud security measures. A good example of the scope of cloud security concerns and issues is seen in the NIST guidelines for cloud security, specified in NIST SP 800-144 (*Guidelines on Security and Privacy in Public Cloud Computing*, December 2011) and listed in Table 13.2. Thus, a full discussion of cloud security is well beyond the scope of this chapter.

### Security Issues for Cloud Computing

Security is important to any computing infrastructure. Companies go to great lengths to secure on-premises computing systems, so it is not surprising that security looms as a major consideration when augmenting or replacing on-premises systems with cloud services. Allaying security concerns is frequently a prerequisite for further discussions about migrating part or all of an organization's computing architecture to the cloud. Availability is another major concern.

Generally speaking, such questions only arise when businesses contemplating moving core transaction processing, such as enterprise resource planning (ERP) systems, and other mission critical applications to the cloud. Companies have traditionally demonstrated less concern about migrating high maintenance applications such as e-mail and payroll to cloud service providers, even though such applications hold sensitive information.

**Table 13.2 NIST Guidelines on Cloud Security and Privacy Issues and Recommendations**

| |
|---|
| **Governance**<br>Extend organizational practices pertaining to the policies, procedures, and standards used for application development and service provisioning in the cloud, as well as the design, implementation, testing, use, and monitoring of deployed or engaged services.<br>Put in place audit mechanisms and tools to ensure organizational practices are followed throughout the system lifecycle. |
| **Compliance**<br>Understand the various types of laws and regulations that impose security and privacy obligations on the organization and potentially impact cloud computing initiatives, particularly those involving data location, privacy and security controls, records management, and electronic discovery requirements.<br>Review and assess the cloud provider's offerings with respect to the organizational requirements to be met and ensure that the contract terms adequately meet the requirements.<br>Ensure that the cloud provider's electronic discovery capabilities and processes do not compromise the privacy or security of data and applications. |
| **Trust**<br>Ensure that service arrangements have sufficient means to allow visibility into the security and privacy controls and processes employed by the cloud provider, and their performance over time.<br>Establish clear, exclusive ownership rights over data.<br>Institute a risk management program that is flexible enough to adapt to the constantly evolving and shifting risk landscape for the lifecycle of the system.<br>Continuously monitor the security state of the information system to support ongoing risk management decisions. |
| **Architecture**<br>Understand the underlying technologies that the cloud provider uses to provision services, including the implications that the technical controls involved have on the security and privacy of the system, over the full system lifecycle and across all system components. |
| **Identity and access management**<br>Ensure that adequate safeguards are in place to secure authentication, authorization, and other identity and access management functions, and are suitable for the organization. |
| **Software isolation**<br>Understand virtualization and other logical isolation techniques that the cloud provider employs in its multi-tenant software architecture, and assess the risks involved for the organization. |
| **Data protection**<br>Evaluate the suitability of the cloud provider's data management solutions for the organizational data concerned and the ability to control access to data; to secure data while at rest, in transit, and in use; and to sanitize data.<br>Take into consideration the risk of collating organizational data with those of other organizations whose threat profiles are high or whose data collectively represent significant concentrated value.<br>Fully understand and weigh the risks involved in cryptographic key management with the facilities available in the cloud environment and the processes established by the cloud provider. |
| **Availability**<br>Understand the contract provisions and procedures for availability, data backup and recovery, and disaster recovery, and ensure that they meet the organization's continuity and contingency planning requirements.<br>Ensure that during an intermediate or prolonged disruption or a serious disaster, critical operations can be immediately resumed, and that all operations can be eventually reinstituted in a timely and organized manner. |

*(Continued)*

Table 13.2 *(Continued)*

| **Incident response** |
| --- |
| Understand the contract provisions and procedures for incident response and ensure that they meet the requirements of the organization. |
| Ensure that the cloud provider has a transparent response process in place and sufficient mechanisms to share information during and after an incident. |
| Ensure that the organization can respond to incidents in a coordinated fashion with the cloud provider in accordance with their respective roles and responsibilities for the computing environment. |

Auditability is another concern for many organizations. For example, in the U.S., many organizations must comply with Sarbanes-Oxley and/or Health and Human Services Health Insurance Portability and Accountability Act (HIPAA) regulations. The auditability of their data must be ensured whether it is stored on premises or moved to the cloud.

Before moving critical infrastructure to the cloud, businesses should perform due diligence on security threats both from outside and inside the cloud. Many of the security issues associated with protecting clouds from outside threats are similar to those that have traditionally faced centralized data centers. In the cloud, however, responsibility for assuring adequate security is frequently shared among users, vendors, and any third-party firms that users rely on for security-sensitive software or configurations. Cloud users are responsible for application-level security. Cloud vendors are responsible for physical security and some software security such as enforcing external firewall policies. Security for intermediate layers of the software stack is shared between users and vendors.

A security risk that should not be overlooked by companies considering a migration to the cloud is that posed by sharing vendor resources with other cloud users. Cloud providers must guard against theft or denial-of-service attacks by their users and users need to be protected from one another. Virtualization can be a powerful mechanism for addressing these potential risks because it protects against most attempts by users to attack one another or the provider's infrastructure. However, not all resources are virtualized, and not all virtualization environments are bug free. Incorrect virtualization may allow user code to access to sensitive portions of the provider's infrastructure or the resources of other users. Once again, these security issues are not unique to the cloud and are similar to those involved in managing non-cloud data centers, where different applications need to be protected from one another.

Another security concern that businesses should consider is the extent to which subscribers are protected against the provider, especially in the area of inadvertent data loss. For example, in the event of provider infrastructure improvements, what happens to hardware that is retired or replaced? It is easy to imagine a hard disk being disposed of without being properly wiped clean of subscriber data. It is also easy to imagine permissions bugs or errors that make subscriber data visible to unauthorized users. User-level encryption may be an important self-help mechanism for subscribers, but businesses should ensure that other protections are in place to avoid inadvertent data loss.

## Addressing Cloud Computing Security Concerns

Numerous documents have been developed to guide business thinking about the security issues associated with cloud computing. In addition to NIST SP 800-144,

**Table 13.3 Control Functions and Classes**

| Technical | Operational | Management |
|---|---|---|
| Access Control<br>Audit and Accountability<br>Identification and Authentication<br>System and Communication Protection | Awareness and Training<br>Configuration and Management<br>Contingency Planning<br>Incident Response<br>Maintenance<br>Media Protection<br>Physical and Environmental Protection<br>Personnel Security System and Information Integrity | Certification, Accreditation and Security Assessment<br>Planning Risk Assessment<br>System and Services Acquisition |

which provides overall guidance, there is also NIST SP 800-146 (*Cloud Computing Synopsis and Recommendations*, May 2012). NIST's recommendations systematically consider each of the major types of cloud services consumed by businesses, including SaaS, IaaS, and PaaS. While security issues vary somewhat depending on the type of cloud service, there are multiple NIST recommendations that are independent of service type. Not surprisingly, NIST recommends selecting cloud providers that support strong encryption, have appropriate redundancy mechanisms in place, employ authentication mechanisms, and offer subscribers sufficient visibility about mechanisms used to protect subscribers from other subscribers and the provider. NIST SP 800-146 also lists the overall security controls that are relevant in a cloud computing environment and that must be assigned to the different cloud actors. These are listed in Table 13.3.

As more businesses incorporate cloud services into their enterprise network infrastructures, cloud computing security will persist as an important issue. Examples of cloud computing security failures have the potential to have a chilling effect on business interest in cloud services. This is inspiring service providers to be serious about incorporating security mechanisms that will allay concerns of potential subscribers. Some service providers have moved their operations to Tier 4 data centers (see Section 5.8) to address user concerns about availability and redundancy. As so many businesses remain reluctant to embrace cloud computing in a big way, cloud service providers will have to continue to work hard to convince potential customers that computing support for core business processes and mission critical applications can be moved safely and securely to the cloud.

## 13.3 CLOUD SECURITY APPROACHES

### Risks and Countermeasures

In general terms, security controls in cloud computing are similar to the security controls in any IT environment. However, because of the operational models and technologies used to enable cloud service, cloud computing may present risks that are specific to the cloud environment. The essential concept in this regard is that while the enterprise loses a substantial amount of control over resources, services, and applications, it must maintain accountability for security and privacy policies.

The Cloud Security Alliance [CSA13] lists the following as the top cloud-specific security threats:

- **Abuse and nefarious use of cloud computing:** For many CSPs, it is relatively easy to register and begin using cloud services, some even offering free limited trial periods. This enables attackers to get inside the cloud to conduct various attacks, such as spamming, malicious code attacks, and denial of service. PaaS providers have traditionally suffered most from this kind of attacks; however, recent evidence shows that hackers have begun to target IaaS vendors as well. The burden is on the CSP to protect against such attacks, but cloud service clients must monitor activity with respect to their data and resources to detect any malicious behavior.

  Countermeasures include (1) stricter initial registration and validation processes; (2) enhanced credit card fraud monitoring and coordination; (3) comprehensive inspection of customer network traffic; and (4) monitoring public blacklists for one's own network blocks.

- **Insecure interfaces and APIs:** CSPs expose a set of software interfaces or APIs that customers use to manage and interact with cloud services. The security and availability of general cloud services is dependent upon the security of these basic APIs. From authentication and access control to encryption and activity monitoring, these interfaces must be designed to protect against both accidental and malicious attempts to circumvent policy.

  Countermeasures include (1) analyzing the security model of CSP interfaces; (2) ensuring that strong authentication and access controls are implemented in concert with encrypted transmission; and (3) understanding the dependency chain associated with the API.

- **Malicious insiders:** Under the cloud computing paradigm, an organization relinquishes direct control over many aspects of security and, in doing so, confers an unprecedented level of trust onto the CSP. One grave concern is the risk of malicious insider activity. Cloud architectures necessitate certain roles that are extremely high risk. Examples include CSP system administrators and managed security service providers.

  Countermeasures include the following: (1) enforce strict supply chain management and conduct a comprehensive supplier assessment; (2) specify human resource requirements as part of legal contract; (3) require transparency into overall information security and management practices, as well as compliance reporting; and (4) determine security breach notification processes.

- **Shared technology issues:** IaaS vendors deliver their services in a scalable way by sharing infrastructure. Often, the underlying components that make up this infrastructure (CPU caches, GPUs, etc.) were not designed to offer strong isolation properties for a multi-tenant architecture. CSPs typically approach this risk by using isolated VMs for individual clients. This approach is still vulnerable to attack, by both insiders and outsiders, and so can only be a part of an overall security strategy.

  Countermeasures include the following: (1) implement security best practices for installation/configuration; (2) monitor environment for unauthorized

changes/activity; (3) promote strong authentication and access control for administrative access and operations; (4) enforce SLAs for patching and vulnerability remediation; and (5) conduct vulnerability scanning and configuration audits.

- **Data loss or leakage:** For many clients, the most devastating impact from a security breach is the loss or leakage of data. We will address this issue in the next section.

  Countermeasures include the following: (1) implement strong API access control; (2) encrypt and protect integrity of data in transit and at rest; (3) analyze data protection at both design and run time; and (4) implement strong key generation, storage and management, and destruction practices.

- **Account or service hijacking:** Account and service hijacking, usually with stolen credentials, remains a top threat. With stolen credentials, attackers can often access critical areas of deployed cloud computing services, allowing them to compromise the confidentiality, integrity, and availability of those services.

  Countermeasures include the following: (1) prohibit the sharing of account credentials between users and services; (2) leverage strong two-factor authentication techniques where possible; (3) employ proactive monitoring to detect unauthorized activity; and (4) understand CSP security policies and SLAs.

- **Unknown risk profile:** In using cloud infrastructures, the client necessarily cedes control to the cloud provider on a number of issues that may affect security. Thus the client must pay attention to and clearly define the roles and responsibilities involved for managing risks. For example, employees may deploy applications and data resources at the CSP without observing the normal policies and procedures for privacy, security, and oversight.

  Countermeasures include (1) disclosure of applicable logs and data; (2) partial/full disclosure of infrastructure details (e.g., patch levels and firewalls); and (3) monitoring and alerting on necessary information.

Similar lists have been developed by the European Network and Information Security Agency [ENIS09] and NIST SP 800-144.

### Data Protection in the Cloud

There are many ways to compromise data. Deletion or alteration of records without a backup of the original content is an obvious example. Unlinking a record from a larger context may render it unrecoverable, as can storage on unreliable media. Loss of an encoding key may result in effective destruction. Finally, unauthorized parties must be prevented from gaining access to sensitive data.

The threat of data compromise increases in the cloud, due to the number of, and interactions between, risks and challenges that are either unique to the cloud or more dangerous because of the architectural or operational characteristics of the cloud environment.

Database environments used in cloud computing can vary significantly. Some providers support a **multi-instance model**, which provide a unique DBMS running on a VM instance for each cloud subscriber. This gives the subscriber complete control over role definition, user authorization, and other administrative tasks related to

security. Other providers support a **multi-tenant model**, which provides a predefined environment for the cloud subscriber that is shared with other tenants, typically through tagging data with a subscriber identifier. Tagging gives the appearance of exclusive use of the instance, but relies on the cloud provider to establish and maintain a sound secure database environment.

Data must be secured while at rest, in transit, and in use, and access to the data must be controlled. The client can employ encryption to protect data in transit, though this involves key management responsibilities for the CSP. The client can enforce access control techniques, but, again, the CSP is involved to some extent depending on the service model used.

For data at rest, the ideal security measure is for the client to encrypt the database and only store encrypted data in the cloud, with the CSP having no access to the encryption key. So long as the key remains secure, the CSP has no ability to decipher the data, although corruption and other denial-of-service attacks remain a risk. The model depicted in Figure 5.9 works equally well when the data is stored in a cloud.

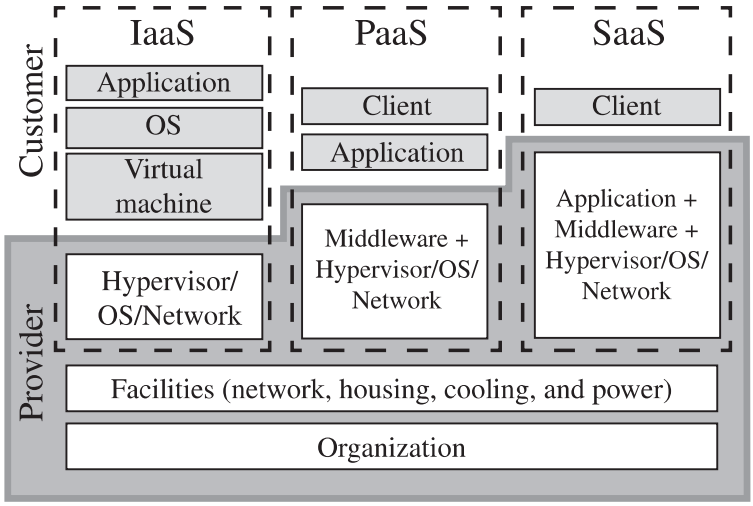
### Security Approaches for Cloud Computing Assets

Beyond the protection and isolation of data, the cloud service provider (CSP) needs to address the broader security considerations for the protection of its assets. Figure 13.5a, adapted from [ENIS15], suggests a categorization of these assets for the three cloud service models. The bottom two layers shown in the figure include organization and facilities. Organization denotes the human resources and the policies and procedures for maintaining the facilities and supporting the delivery of the services. Facilities denote the physical structures and supplies such as networks, cooling, and power supply. Above these levels are the assets specific to the provision of services. For IaaS, the CSP maintains a hypervisor and/or OS on each of its servers, as well as the networking software for interconnection of CSP servers and connection to cloud service consumers (CSCs). Added to these assets for PaaS are the libraries, middleware, and other software to support CSC applications. For SaaS, the CSP also has application software assets for CSC use.

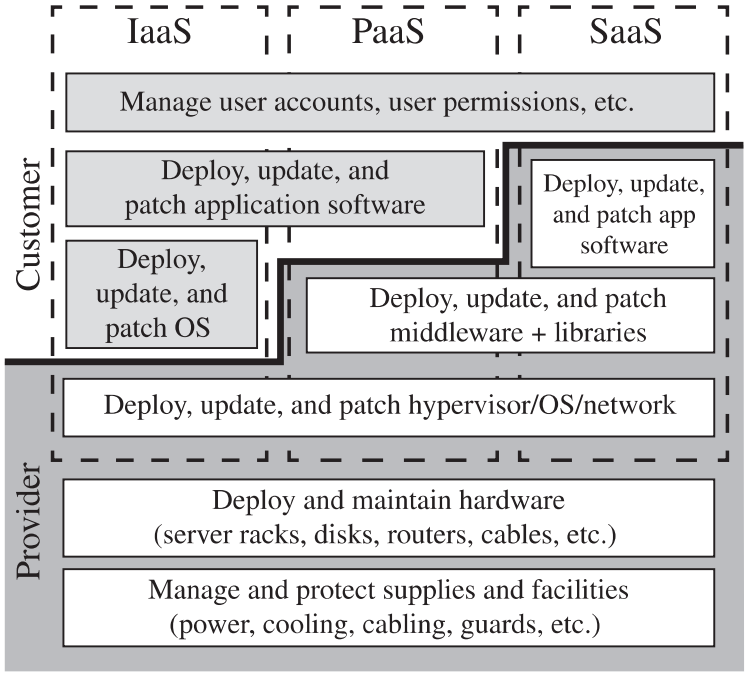
Figure 13.5b suggests key security tasks that are the responsibility of the CSP and of the CSC. The lowest level of the diagram has to do with organizational issues related to the management of its supplies and facilities. These issues will be dealt with in Chapters 14, 15, and 17. The next level of Figure 13.5b covers the physical security of the facility, a topic covered in Chapter 16. Above that, depending on the service model, the CSP is responsible for the security of a range of software capabilities; security measures in the area were addressed in Chapters 11 and 12.

### Cloud Security as a Service

The term **security as a service** has generally meant a package of security services offered by a service provider that offloads much of the security responsibility from an enterprise to the security service provider. Among the services typically provided are authentication, anti-virus, antimalware/spyware, intrusion detection, and security event management. In the context of cloud computing, cloud security as a service, designated SecaaS, is a segment of the SaaS offering of a CSP.




(a) Cloud computing assets




(b) Cloud computing management tasks

**Figure 13.5 Security Considerations for Cloud Computing Assets**

The CSA defines SecaaS as the provision of security applications and services via the cloud either to cloud-based infrastructure and software, or from the cloud to the customers' on-premise systems [CSA11]. The CSA has identified the following SecaaS categories of service:

- Identity and access management
- Data loss prevention
- Web security

- E-mail security
- Security assessments
- Intrusion management
- Security information and event management
- Encryption
- Business continuity and disaster recovery
- Network security

In this section, we examine these categories with a focus on security of the cloud-based infrastructure and services (see Figure 13.6).

**Identity and access management (IAM)** includes people, processes, and systems that are used to manage access to enterprise resources by assuring that the identity of an entity is verified, then granting the correct level of access based on this assured identity. One aspect of identity management is identity provisioning, which has to do with providing access to identified users and subsequently deprovisioning, or denying

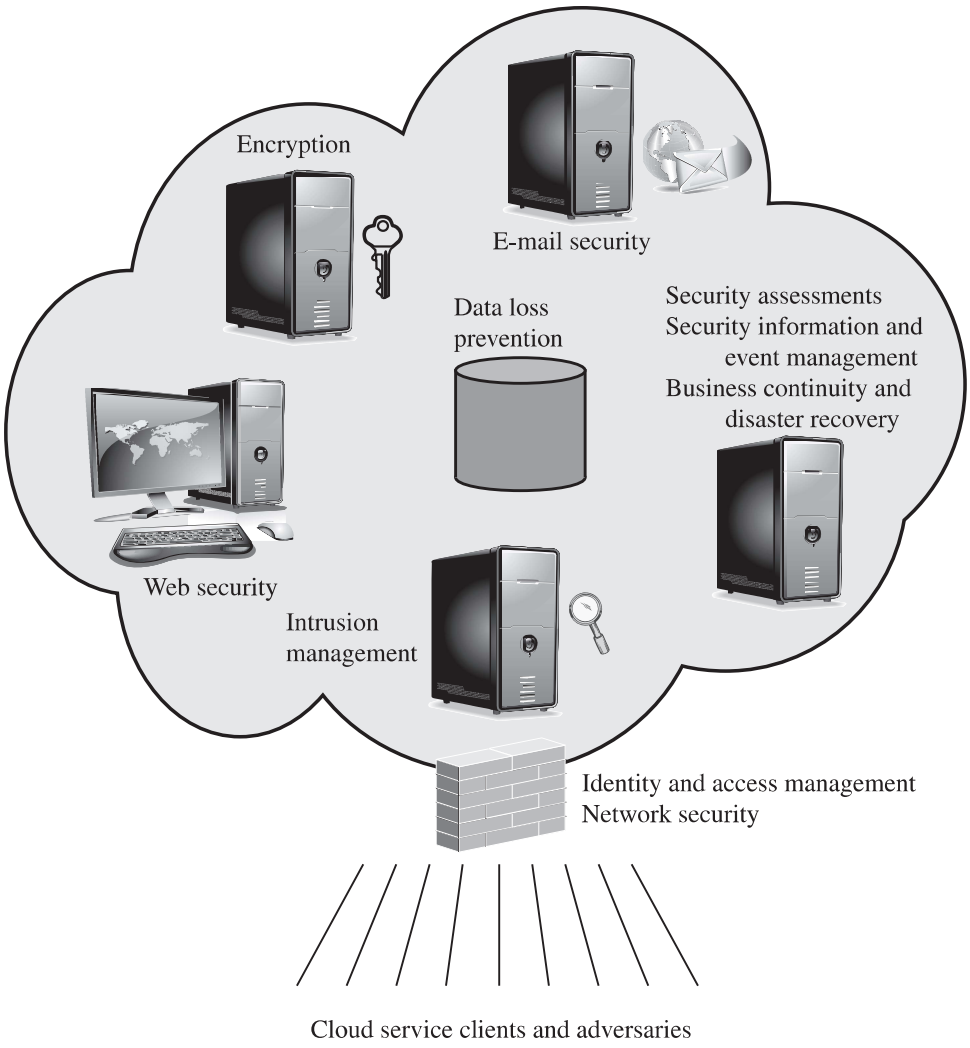



**Figure 13.6 Elements of Cloud Security as a Service**

access, to users when the client enterprise designates such users as no longer having access to enterprise resources in the cloud. Among other requirements, the cloud service provider must be able to exchange identity attributes with the enterprise's chosen identity provider.

The access management portion of IAM involves authentication and access control services. For example, the CSP must be able to authenticate users in a trustworthy manner. The access control requirements in SPI environments include establishing trusted user profile and policy information, using it to control access within the cloud service, and doing this in an auditable way.

**Data loss prevention (DLP)** is the monitoring, protecting, and verifying the security of data at rest, in motion, and in use. Much of DLP can be implemented by the cloud client, such as discussed in previously in this section (Data Protection in the Cloud). The CSP can also provide DLP services, such as implementing rules about what functions can be performed on data in various contexts.

**Web security** is real-time protection offered either on premise through software/appliance installation or via the cloud by proxying or redirecting Web traffic to the CSP. This provides an added layer of protection on top of things like antiviruses to prevent malware from entering the enterprise via activities such as Web browsing. In addition to protecting against malware, a cloud-based Web security service might include usage policy enforcement, data backup, traffic control, and Web access control.

A CSP may provide a Web-based e-mail service, for which security measures are needed. **E-mail security** provides control over inbound and outbound e-mail, protecting the organization from phishing, malicious attachments, enforcing corporate polices such as acceptable use and spam prevention. The CSP may also incorporate digital signatures on all e-mail clients and provide optional e-mail encryption.

**Security assessments** are third-part audits of cloud services. While this service is outside the province of the CSP, the CSP can provide tools and access points to facilitate various assessment activities.

**Intrusion management** encompasses intrusion detection, prevention, and response. The core of this service is the implementation of intrusion detection systems (IDSs) and intrusion prevention systems (IPSs) at entry points to the cloud and on servers in the cloud. An IDS is a set of automated tools designed to detect unauthorized access to a host system. An IPS incorporates IDS functionality and in addition includes mechanisms designed to block traffic from intruders.

**Security information and event management (SIEM)** aggregates (via push or pull mechanisms) log and event data from virtual and real networks, applications, and systems. This information is then correlated and analyzed to provide real-time reporting and alerting on information/events that may require intervention or other type of response. The CSP typically provides an integrated service that can put together information from a variety of sources both within the cloud and within the client enterprise network.

**Encryption** is a pervasive service that can be provided for data at rest in the cloud, e-mail traffic, client-specific network management information, and identity information. Encryption services provided by the CSP involve a range of complex issues, including key management, how to implement virtual private network (VPN) services in the cloud, application encryption, and data content access.

**Business continuity and disaster recovery** comprise measures and mechanisms to ensure operational resiliency in the event of any service interruptions. This is an area where the CSP, because of economies of scale, can offer obvious benefits to a cloud service client. The CSP can provide backup at multiple locations, with reliable failover and disaster recovery facilities. This service must include a flexible infrastructure, redundancy of functions and hardware, monitored operations, geographically distributed data centers, and network survivability.

**Network security** consists of security services that allocate access, distribute, monitor, and protect the underlying resource services. Services include perimeter and server firewalls and denial-of-service protection. Many of the other services listed in this section, including intrusion management, identity and access management, data loss protection, and Web security, also contribute to the network security service.

### An Open-source Cloud Security Module

This section provides an overview of an open-source security module that is part of the OpenStack cloud OS. OpenStack is an open-source software project of the OpenStack Foundation that aims to produce an open-source cloud operating system [ROSA14, SEFR12]. The principal objective is to enable creating and managing huge groups of virtual private servers in a cloud computing environment. OpenStack is embedded, to one degree or another, into data center infrastructure and cloud computing products offered by Cisco, IBM, Hewlett-Packard, and other vendors. It provides multi-tenant IaaS, and aims to meets the needs of public and private clouds regardless of size, by being simple to implement and massively scalable.

The OpenStack OS consists of a number of independent modules, each of which has a project name and a functional name. The modular structure is easy to scale out and provides a commonly used set of core services. Typically, the components are configured together to provide a comprehensive IaaS capability. However, the modular design is such that the components are generally capable of being used independently.

The security module for OpenStack is Keystone. Keystone provides the shared security services essential for a functioning cloud computing infrastructure. It provides the following main services:

- **Identity:** This is user information authentication. This information defines a user's role and permissions within a project, and is the basis for a role-based access control (RBAC) mechanism. Keystone supports multiple methods of authentication, including user name and password, Lightweight Directory Access Protocol (LDAP), and a means of configuring external authentication methods supplied by the CSC.
- **Token:** After authentication, a token is assigned and used for access control. OpenStack services retain tokens and use them to query Keystone during operations.
- **Service catalog:** OpenStack service endpoints are registered with Keystone to create a service catalog. A client for a service connects to Keystone and determines an endpoint to call based on the returned catalog.

- **Policies:** This service enforces different user access levels. Each OpenStack service defines the access policies for its resources in an associated policy file. A resource, for example, could be API access, the ability to attach to a volume, or to fire up instances. These policies can be modified or updated by the cloud administrator to control the access to the various resources.

Figure 13.7 illustrates the way in which Keystone interacts with other OpenStack components to launch a new VM. Nova is the management software module that controls VMs within the IaaS cloud computing platform. It manages the lifecycle of compute instances in an OpenStack environment. Responsibilities include spawning, scheduling, and decommissioning of machines on demand. Thus, Nova enables enterprises and service providers to offer on-demand computing resources by provisioning and managing large networks of VMs. Glance is a lookup and retrieval system for VM disk images. It provides services for discovering, registering, and retrieving virtual images through an API. Swift is a distributed object store that creates a redundant and scalable storage space of up to multiple petabytes of data. Object storage does not present a traditional file system, but rather a distributed storage system for static data such as VM images, photo storage, e-mail storage, backups, and archives.

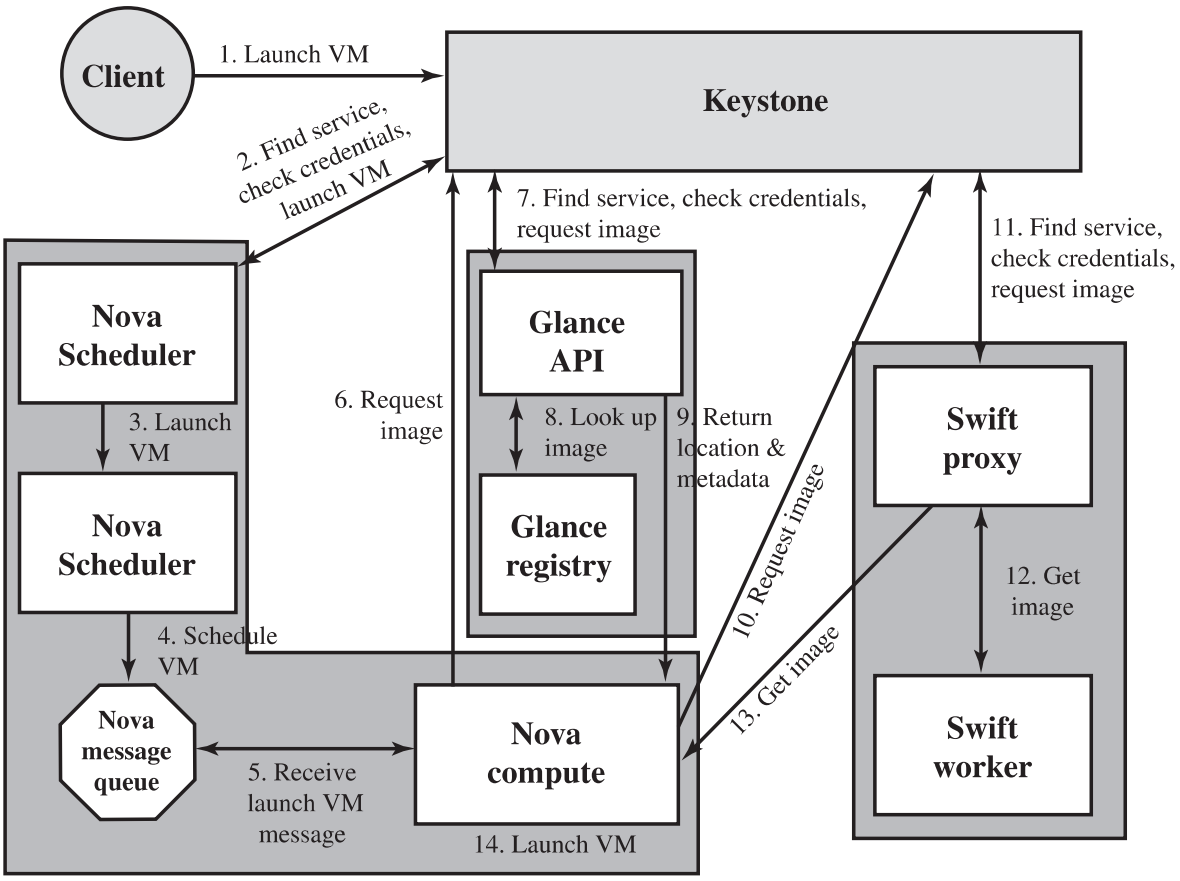



**Figure 13.7 Launching a Virtual Machine in OpenStack**

## 13.4 THE INTERNET OF THINGS

The Internet of things is the latest development in the long and continuing revolution of computing and communications. Its size, ubiquity, and influence on everyday lives, business, and government dwarf any technical advance that has gone before. This section provides a brief overview of the Internet of things.

### Things on the Internet of Things

The Internet of things (IoT) is a term that refers to the expanding interconnection of smart devices, ranging from appliances to tiny sensors. A dominant theme is the embedding of short-range mobile transceivers into a wide array of gadgets and everyday items, enabling new forms of communication between people and things, and between things themselves. The Internet now supports the interconnection of billions of industrial and personal objects, usually through cloud systems. The objects deliver sensor information, act on their environment, and in some cases modify themselves, to create overall management of a larger system, like a factory or city.

The IoT is primarily driven by deeply embedded devices. These devices are low-bandwidth, low-repetition data capture, and low-bandwidth data-usage appliances that communicate with each other and provide data via user interfaces. Embedded appliances, such as high-resolution video security cameras, video VoIP phones, and a handful of others, require high-bandwidth streaming capabilities. Yet countless products simply require packets of data to be intermittently delivered.

### Evolution

With reference to the end systems supported, the Internet has gone through roughly four generations of deployment culminating in the IoT:

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

The fourth generation is usually thought of as the IoT, and which is marked by using billions of embedded devices.

### Components of IoT-enabled Things

The key components of an IoT-enabled device are the following (see Figure 13.8):

- **Sensor:** A sensor measures some parameter of a physical, chemical, or biological entity and delivers an electronic signal proportional to the observed

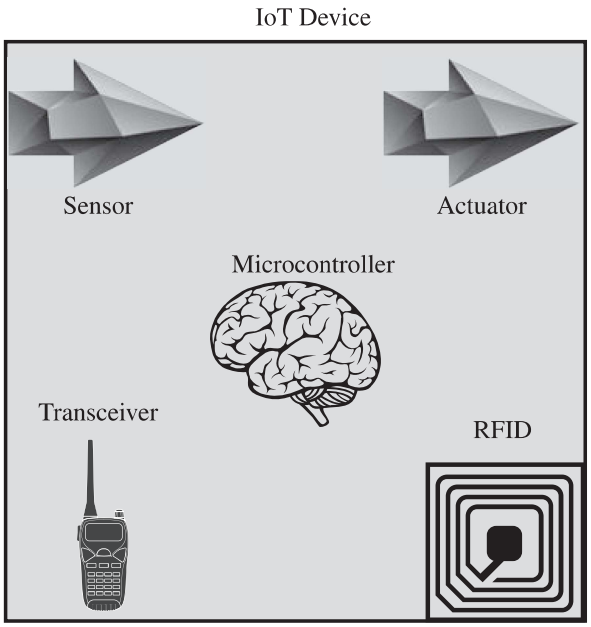



**Figure 13.8** **IoT Components**

characteristic, either in the form of an analog voltage level or a digital signal. In both cases, the sensor output is typically input to a microcontroller or other management element.

- **Actuator:** An actuator receives an electronic signal from a controller and responds by interacting with its environment to produce an effect on some parameter of a physical, chemical, or biological entity.
- **Microcontroller:** The "smart" in a smart device is provided by a deeply embedded microcontroller.
- **Transceiver:** A transceiver contains the electronics needed to transmit and receive data. Most IoT devices contain a wireless transceiver, capable of communication using Wi-Fi, ZigBee, or some other wireless scheme.
- **Radio-frequency Identification (RFID):** (RFID) technology, which uses radio waves to identify items, is increasingly becoming an enabling technology for IoT. The main elements of an RFID system are tags and readers. RFID tags are small programmable devices used for object, animal, and human tracking. They come in a variety of shapes, sizes, functionalities, and costs. RFID readers acquire and sometimes rewrite information stored on RFID tags that come within operating range (a few inches up to several feet). Readers are usually connected to a computer system that records and formats the acquired information for further uses.

### IoT and Cloud Context

To better understand the function of an IoT, it is useful to view it in the context of a complete enterprise network that includes third-party networking and cloud computing elements. Figure 13.9 provides an overview illustration.

*EDGE* At the edge of a typical enterprise network is a network of IoT-enabled devices, consisting of sensors and perhaps actuators. These devices may communicate

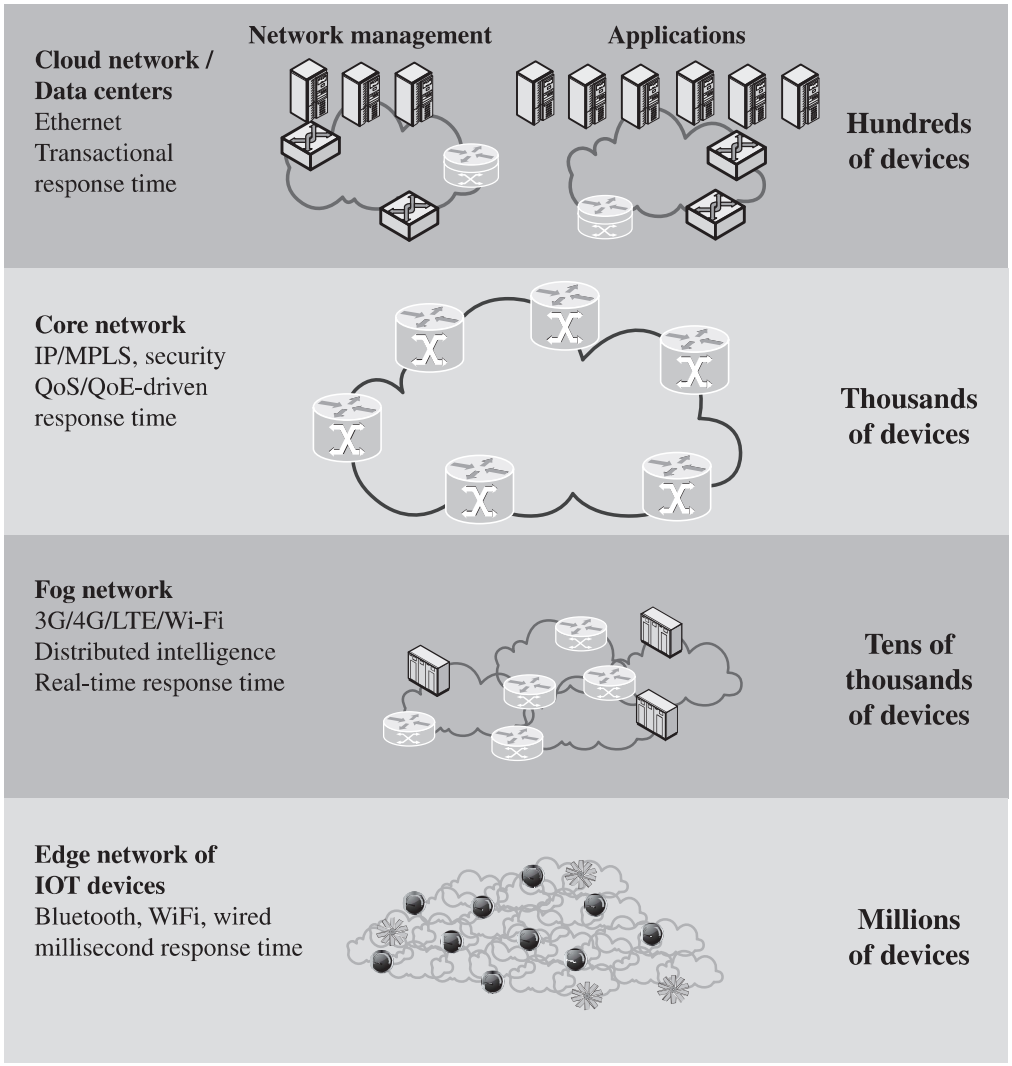



**Figure 13.9** **The IoT and Cloud Context**

with one another. For example, a cluster of sensors may all transmit their data to one sensor that aggregates the data to be collected by a higher-level entity. At this level, there may also be a number of **gateways**. A gateway interconnects the IoT-enabled devices with the higher-level communication networks. It performs the necessary translation between the protocols used in the communication networks and those used by devices. A gateway may also perform a basic data aggregation function.

***Fog*** In many IoT deployments, massive amounts of data may be generated by a distributed network of sensors. For example, offshore oil fields and refineries can generate a terabyte of data per day. An airplane can create multiple terabytes of data per hour. Rather than store all of that data permanently (or at least for a long period) in central storage accessible to IoT applications, it is often desirable to do as much data processing close to the sensors as possible. Thus, the purpose of what is sometimes referred to as the edge computing level is to convert network data flows into information that is suitable for storage

and higher-level processing. Processing elements at these levels may deal with high volumes of data and perform data transformation operations, resulting in the storage of much lower volumes of data. The following are examples of fog computing operations:

- **Evaluation:** Evaluating data for criteria as to whether it should be processed at a higher level.
- **Formatting:** Reformatting data for consistent higher-level processing.
- **Expanding/decoding:** Handling cryptic data with additional context (such as the origin).
- **Distillation/reduction:** Reducing and/or summarizing data to minimize the impact of data and traffic on the network and higher-level processing systems.
- **Assessment:** Determining whether data represent a threshold or alert; this could include redirecting data to additional destinations.

Generally, fog computing devices are deployed physically near the edge of the IoT network; that is, near the sensors and other data-generating devices. Thus, some of the basic processing of large volumes of generated data is offloaded and outsourced from IoT application software located at the center of the network.

Fog computing and fog services are becoming a distinguishing characteristic of the IoT. Fog computing represents an opposite trend in modern networking from cloud computing. With cloud computing, massive, centralized storage and processing resources are made available to distributed customers over cloud networking facilities to a relatively small number of users. With fog computing, massive numbers of individual smart objects are interconnected with fog networking facilities that provide processing and storage resources close to the edge devices in an IoT. Fog computing addresses the challenges raised by the activity of thousand or millions of smart devices, including security, privacy, network capacity constraints, and latency requirements. The term *fog computing* is inspired by the fact that fog tends to hover low to the ground, whereas clouds are high in the sky.

***Core*** The core network, also referred to as a **backbone network**, connects geographically dispersed fog networks as well as provides access to other networks that are not part of the enterprise network. Typically, the core network will use very high performance routers, high-capacity transmission lines, and multiple interconnected routers for increased redundancy and capacity. The core network may also connect to high-performance, high-capacity servers, such as large database servers and private cloud facilities. Some of the core routers may be purely internal, providing redundancy and additional capacity without serving as edge routers.

***Cloud*** The cloud network provides storage and processing capabilities for the massive amounts of aggregated data that originate in IoT-enabled devices at the edge. Cloud servers also host the applications that (1) interact with and manage the IoT devices, and (2) analyze the IoT-generated data. Table 13.4 compares cloud and fog computing.

**Table 13.4 Comparison of Cloud and Fog Features**

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

## 13.5 IOT SECURITY

IoT is perhaps the most complex and undeveloped area of network security. To see this, consider Figure 13.10, which shows the main elements of interest for IoT security. At the center of the network are the application platforms, data storage servers, and network and security management systems. These central systems gather data from sensors, send control signals to actuators, and are responsible for managing the IoT devices and their communication networks. At the edge of the network are IoT-enabled devices, some of which are quite simple constrained devices, and some of which are more intelligent unconstrained devices. As well, gateways may perform protocol conversion and other networking service on behalf of IoT devices.

Figure 13.10 illustrates a number of typical scenarios for interconnection and the inclusion of security features. The shading in Figure 13.10 indicates the systems that support at least some of these functions. Typically, gateways will implement secure functions, such as TLS and IPsec. Unconstrained devices may or may not implement some security capability. Constrained devices generally have limited or no security features. As suggested in the figure, gateway devices can provide secure communication between the gateway and the devices at the center, such as application platforms and management platforms. However, any constrained or unconstrained devices attached to the gateway are outside the zone of security established between the gateway and the central systems. As shown, unconstrained devices can communicate directly with the center and support security functions. However, constrained devices that are not connected to gateways have no secure communications with central devices.

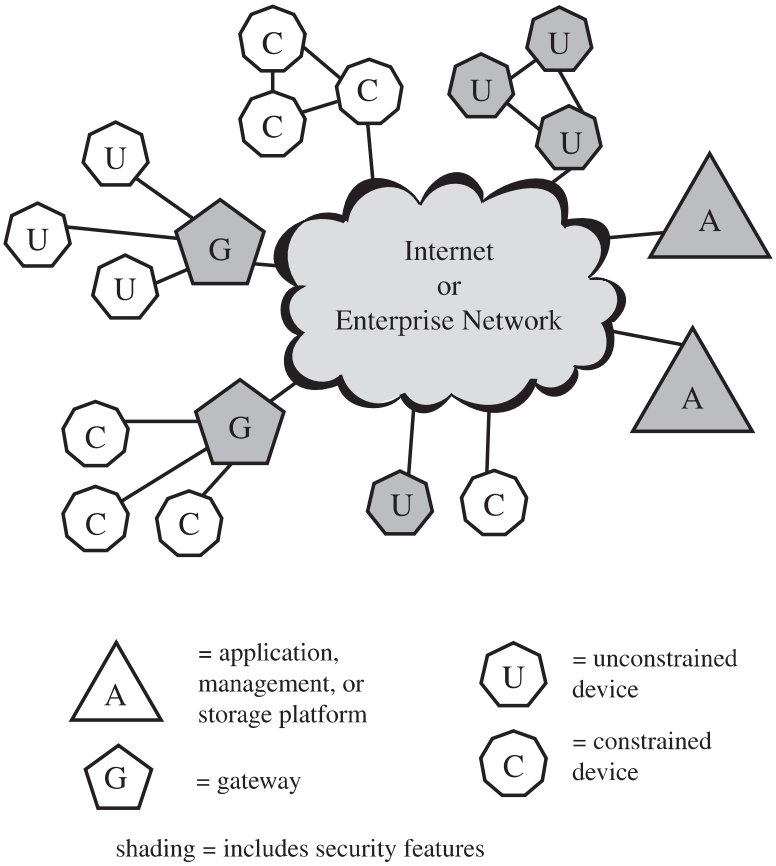



**Figure 13.10** **IoT Security: Elements of Interest**

### The Patching Vulnerability

In an often-quoted 2014 article, security expert Bruce Schneier stated that we are at a crisis point with regard to the security of embedded systems, including IoT devices [SCHN14]. The embedded devices are riddled with vulnerabilities and there is no good way to patch them. The chip manufacturers have strong incentives to produce their product with its firmware and software as quickly and cheaply as possible. The device manufacturers choose a chip based on price and features and do very little if anything to the chip software and firmware. Their focus is the functionality of the device itself. The end user may have no means of patching the system or, if so, little information about when and how to patch. The result is that the hundreds of millions of Internet-connected devices in the IoT are vulnerable to attack. This is certainly a problem with sensors, allowing attackers to insert false data into the network. It is potentially a graver threat with actuators, where the attacker can affect the operation of machinery and other devices.

### IoT Security and Privacy Requirements Defined by ITU-T

ITU-T Recommendation Y.2066 (*Common Requirements of the Internet of Things*, June 2014) includes a list of security requirements for the IoT. This list is a useful baseline for understanding the scope of security implementation needed for an IoT deployment. The requirements are defined as being the functional requirements

during capturing, storing, transferring, aggregating, and processing the data of things, as well as to the provision of services which involve things. These requirements are related to all the IoT actors. The requirements are the following:

- **Communication security:** Secure, trusted, and privacy protected communication capability is required, so unauthorized access to the content of data can be prohibited, integrity of data can be guaranteed and privacy-related content of data can be protected during data transmission or transfer in IoT.
- **Data management security:** Secure, trusted, and privacy protected data management capability is required, so unauthorized access to the content of data can be prohibited, integrity of data can be guaranteed, and privacy-related content of data can be protected when storing or processing data in IoT.
- **Service provision security:** Secure, trusted, and privacy protected service provision capability is required, so unauthorized access to service and fraudulent service provision can be prohibited and privacy information related to IoT users can be protected.
- **Integration of security policies and techniques:** The ability to integrate different security policies and techniques is required, so as to ensure a consistent security control over the variety of devices and user networks in IoT.
- **Mutual authentication and authorization:** Before a device (or an IoT user) can access the IoT, mutual authentication and authorization between the device (or the IoT user) and IoT is required to be performed according to predefined security policies.
- **Security audit:** Security audit is required to be supported in IoT. Any data access or attempt to access IoT applications are required to be fully transparent, traceable and reproducible according to appropriate regulation and laws. In particular, IoT is required to support security audit for data transmission, storage, processing, and application access.

A key element in providing security in an IoT deployment is the gateway. ITU-T Recommendation Y.2067 (*Common Requirements and Capabilities of a Gateway for Internet of Things Applications*, June 2014) details specific security functions that the gateway should implement, some of which are illustrated in Figure 13.11. These consist of the following:

- Support identification of each access to the connected devices.
- Support authentication with devices. Based on application requirements and device capabilities, it is required to support mutual or one-way authentication with devices. With one-way authentication, either the device authenticates itself to the gateway or the gateway authenticates itself to the device, but not both.
- Support mutual authentication with applications.
- Support the security of the data that are stored in devices and the gateway, or transferred between the gateway and devices, or transferred between the gateway and applications. Support the security of these data based on security levels.
- Support mechanisms to protect privacy for devices and the gateway.

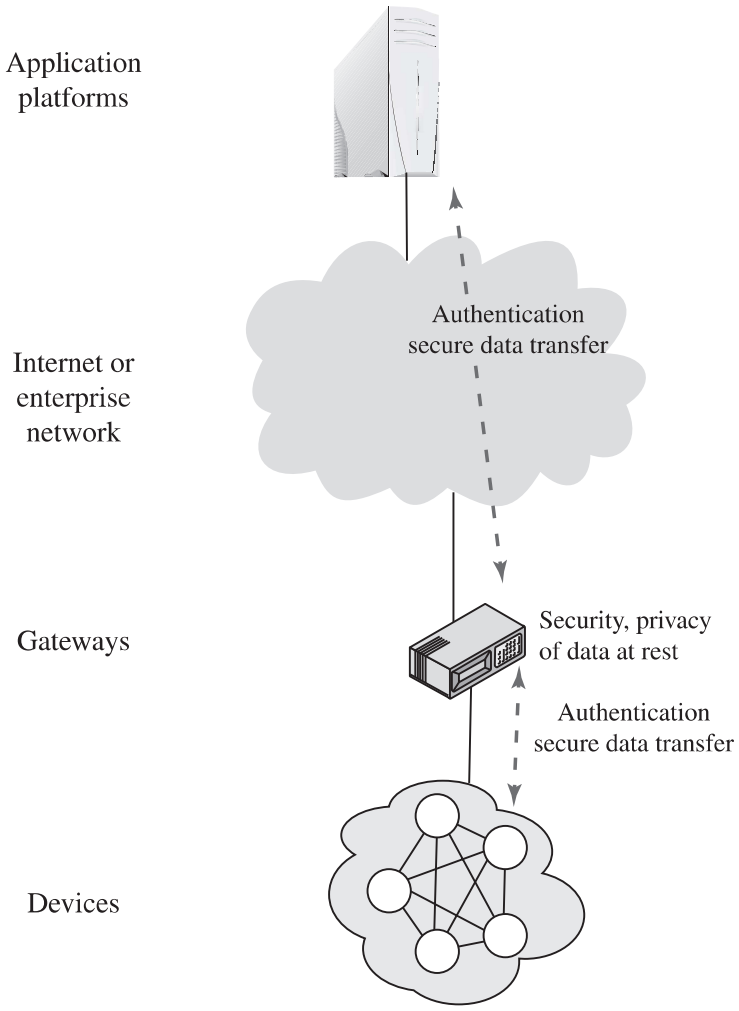



**Figure 13.11 IoT Gateway Security Functions**

- Support self-diagnosis and self-repair as well as remote maintenance.
- Support firmware and software update.
- Support auto configuration or configuration by applications. The gateway is required to support multiple configuration modes, for example, remote and local configuration, automatic and manual configuration, and dynamic configuration based on policies.

Some of these requirements may be difficult to achieve when they involve providing security services for constrained devices. For example, the gateway should support security of data stored in devices. Without encryption capability at the constrained device, this may be impractical to achieve.

Note the Y.2067 requirements make a number of references to privacy requirements. Privacy is an area of growing concern with the widespread deployment of IoT-enabled things in homes, retail outlets, and vehicles and humans. As more things are interconnected, governments and private enterprises will collect massive amounts of data about individuals, including medical information, location and movement information, and application usage.

## An IoT Security Framework

Cisco has developed a framework for IoT security [FRAH15] that serves as a useful guide to the security requirements for IoT. Figure 13.12 illustrates the security environment related to the logical structure of an IoT. The IoT model is a simplified version of the World Forum IoT Reference Model. It consists of the following levels:

- **Smart objects/embedded systems:** Consists of sensors, actuators, and other embedded systems at the edge of the network. This is the most vulnerable part of an IoT. The devices may not be in a physically secure environment and may need to function for years. Availability is certainly an issue. Network managers also need to be concerned about the authenticity and integrity of the data generated by sensors and about protecting actuators and other smart devices from unauthorized use. Privacy and protection from eavesdropping may also be requirements.
- **Fog/edge network:** This level is concerned with the wired and wireless interconnection of IoT devices. In addition, a certain amount of data processing and consolidation may be done at this level. A key issue of concern is the wide variety of network technologies and protocols used by the various IoT devices and the need to develop and enforce a uniform security policy.
- **Core network:** The core network level provides data paths between network center platforms and the IoT devices. The security issues here are those confronted in traditional core networks. However, the vast number of endpoints to interact with and manage creates a substantial security burden.
- **Data center/cloud:** This level contains the application, data storage, and network management platforms. IoT does not introduce any new security issues at this level, other than the necessity of dealing with huge numbers of individual endpoints.

Within this four-level architecture, the Cisco model defines four general security capabilities that span multiple levels:

- **Role-based security:** RBAC systems assign access rights to roles instead of individual users. In turn, users are assigned to different roles, either statically

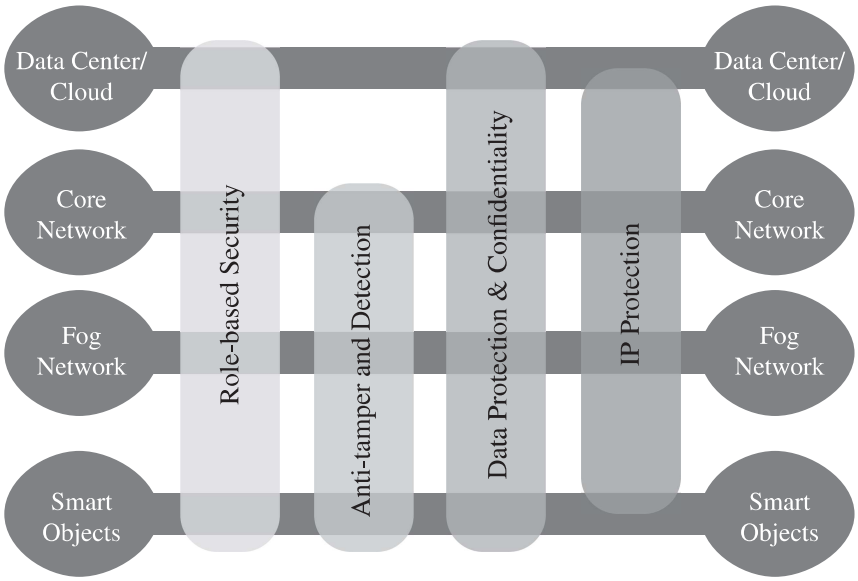



Figure 13.12 **IoT Security Environment**

or dynamically, according to their responsibilities. RBAC enjoys widespread commercial use in cloud and enterprise systems and is a well-understood tool that can be used to manage access to IoT devices and the data they generate.

- **Anti-tamper and detection:** This function is particularly important at the device and fog network levels but also extends to the core network level. All of these levels may involve components that are physically outside the area of the enterprise that is protected by physical security measures.
- **Data protection and confidentiality:** These functions extend to all level of the architecture.
- **Internet protocol protection:** Protection of data in motion from eavesdropping and snooping is essential between all levels.

Figure 13.12 maps specific security functional areas across the four layers of the IoT model. [FRAH15] also proposes a secure IoT framework that defines the components of a security facility for an IoT that encompasses all the levels, as shown in Figure 13.13. The four components are:

- **Authentication:** Encompasses the elements that initiate the determination of access by first identifying the IoT devices. In contrast to typical enterprise network devices, which may be identified by a human credential (e.g., username and password or token), the IoT endpoints must be fingerprinted by means that do not require human interaction. Such identifiers include RFID, x.509 certificates, or the MAC address of the endpoint.
- **Authorization:** Controls a device's access throughout the network fabric. This element encompasses access control. Together with the authentication layer, it establishes the necessary parameters to enable the exchange of information

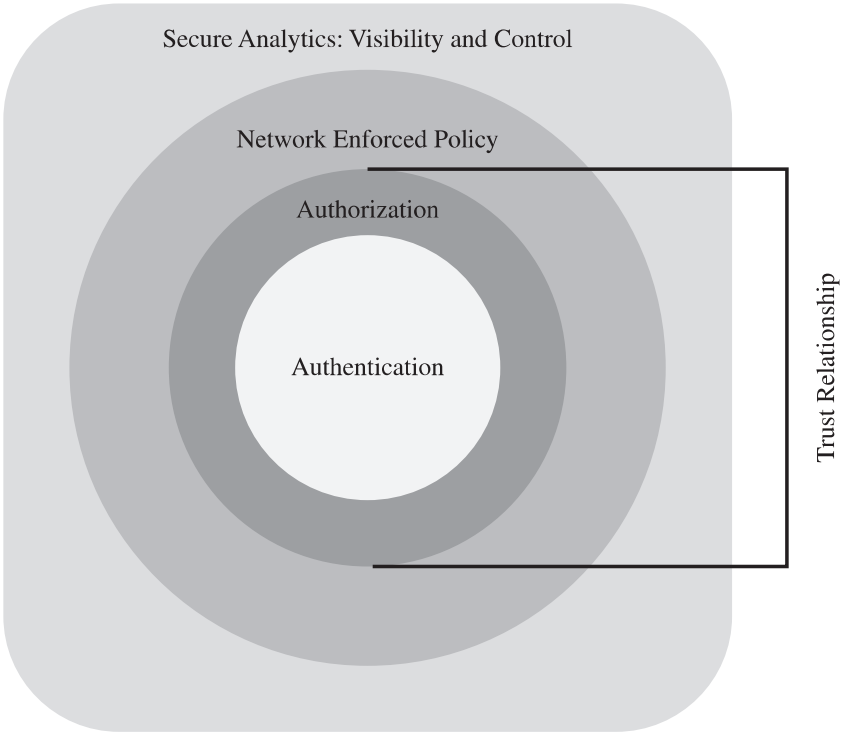



Figure 13.13 **Secure IoT Framework**

between devices and between devices and application platforms and enables IoT-related services to be performed.

- **Network enforced policy:** Encompasses all elements that route and transport endpoint traffic securely over the infrastructure, whether control, management, or actual data traffic.
- **Secure analytics, including visibility and control:** This component includes all the functions required for central management of IoT devices. This involves, firstly, visibility of IoT devices, which simply means that central management services are securely aware of the distributed IoT device collection, including identity and attributes of each device. Building on this visibility is the ability to exert control, including configuration, patch updates, and threat countermeasures.

An important concept related to this framework is that of trust relationship. In this context, trust relationship refers to the ability of the two partners to an exchange to have confidence in the identity and access rights of the other. The authentication component of the trust framework provides a basic level of trust, which is expanded with the authorization component. [FRAH15] gives the example that a car may establish a trust relationship with another car from the same vendor. That trust relationship, however, may only allow cars to exchange their safety capabilities. When a trusted relationship is established between the same car and its dealer's network, the car may be allowed to share additional information such as its odometer reading and last maintenance record.

### An Open-source IoT Security Module

This section provides an overview of MiniSec, an open-source security module that is part of the TinyOS operating system. TinyOS is designed for small embedded systems with tight requirements on memory, processing time, real-time response, and power consumption. TinyOS takes the process of streamlining quite far, resulting in a very minimal OS for embedded systems, with a typical configuration requiring 48 KB of code and 10 KB of RAM [LEVI12]. The main application of TinyOS is wireless sensor networks, and it has become the de facto OS for such networks. With sensor networks the primary security concerns relate to wireless communications. MiniSec is designed to be a link-level module that offers a high level of security, while simultaneously keeping energy consumption low and using very little memory [LUK07]. MiniSec provides confidentiality, authentication, and replay protection.

MiniSec has two operating modes, one tailored for single-source communication, and another tailored for multi-source broadcast communication. The latter does not require per-sender state for replay protection and thus scales to large networks.

MiniSec is designed to meet the following requirements:

- **Data authentication:** Enables a legitimate node to verify whether a message originated from another legitimate node (i.e., a node with which it shares a secret key) and was unchanged during transmission.
- **Confidentiality:** A basic requirement for any secure communications system.
- **Replay protection:** Prevents an attacker from successfully recording a packet and replaying it at a later time.
- **Freshness:** Because sensor nodes often stream time-varying measurements, providing guarantee of message freshness is an important property. There are

two types of freshness: Strong and weak. MiniSec provides a mechanism to guarantee weak freshness, where a receiver can determine a partial ordering over received messages without a local reference time point.

- **Low energy overhead:** This is achieved by minimizing communication overhead and by using only symmetric encryption.
- **Resilient to lost messages:** The relatively high occurrence of dropped packets in wireless sensor networks requires a design that can tolerate high message loss rates.

*Cryptographic Algorithms* Two cryptographic algorithms used by MiniSec are worth noting. The first of these is the encryption algorithm Skipjack. Skipjack was developed in the 1990s by the U.S. National Security Agency (NSA). It is one of the simplest and fastest block cipher algorithms, which is critical to embedded systems. A study of eight possible candidate algorithms for wireless security networks [LAW06] concluded that Skipjack was the best algorithm in terms of code memory, data memory, encryption/decryption efficiency, and key setup efficiency.

Skipjack makes use of an 80-bit key. It was intended by NSA to provide a secure system once it became clear that DES, with only a 56-bit key, was vulnerable. Contemporary algorithms, such as AES, employ a key length of at least 128 bits, and 80 bits is generally considered inadequate. However, for the limited application of wireless sensor networks and other IoT devices, which provide large volumes of short data blocks over a slow data link, Skipjack suffices. With its efficient computation and low memory footprint, Skipjack is an attractive choice for IoT devices.

The block cipher mode of operation chosen for MiniSec is the Offset Codebook (OCB) mode. As mentioned in Chapter 2, a mode of operation must be specified when a plaintext source consists of multiple blocks of data to be encrypted with the same encryption key. OCB mode is provably secure assuming the underlying block cipher is secure. OCB mode is a one-pass mode of operation making it highly efficient. Only one block cipher call is necessary for each plaintext block, (with an additional two calls needed to complete the whole encryption process). OCB is especially well suited for the stringent energy constraints of sensor nodes.

A feature that contributes significantly to the efficiency of OCB is that with one pass through the sequence of plaintext blocks, it produces a ciphertext of equal length and a tag for authentication. To decrypt a ciphertext, the receiver performs the reverse process to recover the plaintext. Then, the receiver ensures that the tag is as expected. If the receiver computes a different tag than the one accompanying the ciphertext, the ciphertext is considered to be invalid. Thus, both message authentication and message confidentiality are achieved with a single, simple algorithm. OCB will be described in Chapter 21.

MiniSec employs per-device keys; that is, each key is unique to a particular pair of devices to prevent replay attacks.

*Operating Modes* MiniSec has two operating modes: Unicast (MiniSec-U) and broadcast (MiniSec-B). Both schemes use OCB with a counter, known as a nonce, that is input along with the plaintext into the encryption algorithm. The least significant bits of the counter are also sent as plaintext to enable synchronization. For both modes, data are transmitted in packets. Each packet includes the encrypted data block, the OCB authentication tag, and the MiniSec counter.

MiniSec-U employs synchronized counters, which require the receiver to keep a local counter for each sender. The strictly monotonically increasing counter guarantees semantic confidentiality.[1] Even if the sender *A* repeatedly sends the same message, each ciphertext is different because a different counter value is used. In addition, once a receiver observes a counter value, it rejects packets with an equal or smaller counter value. Therefore, an attacker cannot replay any packet that the receiver has previously received. If a number of packets are dropped, the sender and receiver engage in a resynchronization protocol.

MiniSec-U cannot be directly used to secure broadcast communication. First, it would be too expensive to run the counter resynchronization protocol among many receivers. In addition, if a node was to simultaneously receive packets from a large group of sending nodes, it would need to maintain a counter for each sender, resulting in high memory overhead. Instead, it uses two mechanisms, a timing-based approach and a bloom-filter approach, that defend against replay attacks. First, the time is divided into *t*-length epochs $E1, E2, \ldots$. Using the current epoch or the previous epoch as nonce for OCB encryption, the replay of messages from older epochs is avoided. The timing approach is augmented with a bloom-filter approach in order to prevent replay attacks inside the current epoch. MiniSec-B uses as nonce element in OCB encryption and bloom-filter key the string *nodeID.Ei.Cab*, where *nodeID* is the sender node identifier, *Ei* is the current epoch, and *Cab* is a shared counter. Every time that a node receives a message, it checks if it belongs to its bloom filter. If the message is not replayed, it is stored in the bloom filter. Else, the node drops it.

For further details on the two operating modes, see [TOBA07].

## 13.6 KEY TERMS AND REVIEW QUESTIONS

### Key Terms

| | | |
|---|---|---|
| actuator | identity and access management (IAM) | public cloud |
| backbone network | infrastructure as a service (IaaS) | radio-frequency identification (RFID) |
| cloud auditor | Internet of things (IoT) | security as a service (SecaaS) |
| cloud broker | intrusion management | security assessments |
| cloud carrier | microcontroller | security information and event management (SIEM) |
| cloud computing | MiniSec | sensor |
| cloud service consumer (CSC) | multi-instance model | service arbitrage |
| cloud service provider (CSP) | multi-tenant model | service aggregation |
| community cloud | OpenStack | service intermediation |
| core | patching vulnerability | software as a service (SaaS) |
| data loss prevention (DLP) | platform as a service (PaaS) | transceiver |
| edge | private cloud | |
| fog | | |
| hybrid cloud | | |

[1] Semantic confidentiality means that if the same plaintext is encrypted twice, the two resulting ciphertexts are different.

## Review Questions

13.1 List five essential characteristics of cloud computing.
13.2 List and briefly define three cloud service models.
13.3 Briefly explain the most prominent deployment models for cloud computing.
13.4 Describe some of the main cloud-specific security threats.
13.5 What is OpenStack?
13.6 Define the Internet of things.
13.7 List any five security recommendations included in the ITU-T recommendation.
13.8 Define the patching vulnerability.
13.9 What is the IoT security framework?
13.10 What are some of the key features of the Skipjack encryption algorithm?