

# Week 7

---

SOFTWARE SECURITY

OPERATING SYSTEM SECURITY

CLOUD AND IOT SECURITY

# Chapter 11

## Software Security

# CWE/SANS Top 25 Most Dangerous Software Errors (2011)

---

## **Software Error Category: Insecure Interaction Between Components**

Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')  
Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')  
Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')  
Unrestricted Upload of File with Dangerous Type  
Cross-Site Request Forgery (CSRF)  
URL Redirection to Untrusted Site ('Open Redirect')

## **Software Error Category: Risky Resource Management**

Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')  
Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')  
Download of Code Without Integrity Check  
Inclusion of Functionality from Untrusted Control Sphere  
Use of Potentially Dangerous Function  
Incorrect Calculation of Buffer Size  
Uncontrolled Format String  
Integer Overflow or Wraparound

## **Software Error Category: Porous Defenses**

Missing Authentication for Critical Function  
Missing Authorization  
Use of Hard-coded Credentials  
Missing Encryption of Sensitive Data  
Reliance on Untrusted Inputs in a Security Decision  
Execution with Unnecessary Privileges  
Incorrect Authorization  
Incorrect Permission Assignment for Critical Resource  
Use of a Broken or Risky Cryptographic Algorithm  
Improper Restriction of Excessive Authentication Attempts  
Use of a One-Way Hash without a Salt

# Security Flaws

---

These flaws occur as a consequence of insufficient checking and validation of data and error codes in programs

Awareness of these issues is a critical initial step in writing more secure program code

Emphasis should be placed on the need for software developers to address these known areas of concern

Critical Web application security flaws include five related to insecure software code

- Unvalidated input
- Cross-site scripting
- Buffer overflow
- Injection flaws
- Improper error handling

# Reducing Software Vulnerabilities

---

The NIST report NISTIR 8151 presents a range of approaches to reduce the number of software vulnerabilities

It recommends:

- Stopping vulnerabilities before they occur by using improved methods for specifying and building software
- Finding vulnerabilities before they can be exploited by using better and more efficient testing techniques
- Reducing the impact of vulnerabilities by building more resilient architectures

# Software Quality/Reliability vs Security

---

## Software quality and reliability:

- Concerned with the accidental failure of program as a result of some theoretically random, unanticipated input, system interaction, or use of incorrect code
- Improve using structured design and testing to identify and eliminate as many bugs as possible from a program
- Concern is not how many bugs, but how often they are triggered

## Software security:

- Attacker chooses probability distribution, specifically targeting bugs that result in a failure that can be exploited by the attacker
- Triggered by inputs that differ dramatically from what is usually expected
- Unlikely to be identified by common testing approaches

# Defensive Programming

---

- ▢ Also referred to as secure programming
- ▢ Designing and implementing software so that it continues to function even when under attack
- ▢ Requires attention to all aspects of program execution, environment, and type of data it processes
- ▢ Software is able to detect erroneous conditions resulting from some attack
- ▢ Key rule is to never assume anything, check all assumptions and handle any possible error states

# Defensive Programming - Challenges

---

Programmers often make assumptions about the type of inputs a program will receive and the environment it executes in

Assumptions need to be validated by the program and all potential failures handled gracefully and safely

Requires a changed mindset to traditional programming practices

Programmers have to understand how failures can occur and the steps needed to reduce the chance of them occurring in their programs

Conflicts with business pressures to keep development times as short as possible to maximize market advantage



# Defensive Programming - Topics

---

Input validation

---

Correct data interpretation

---

Handling of race conditions

---

Shell script execution vulnerabilities

---

Use of privileges (root)

---

File I/O

---

Inter-program communications

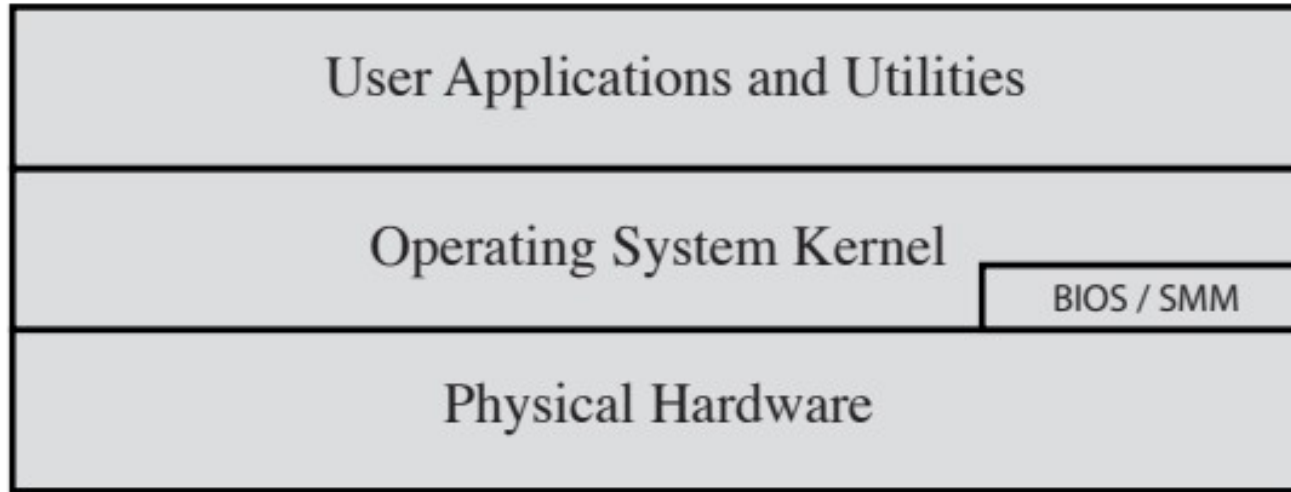
---

Use of third-party software and libraries

---

## Chapter 12

# Operating System Security



**Figure 12.1 Operating System Security Layers**

# OS Security Layers

---

# Operating System Security

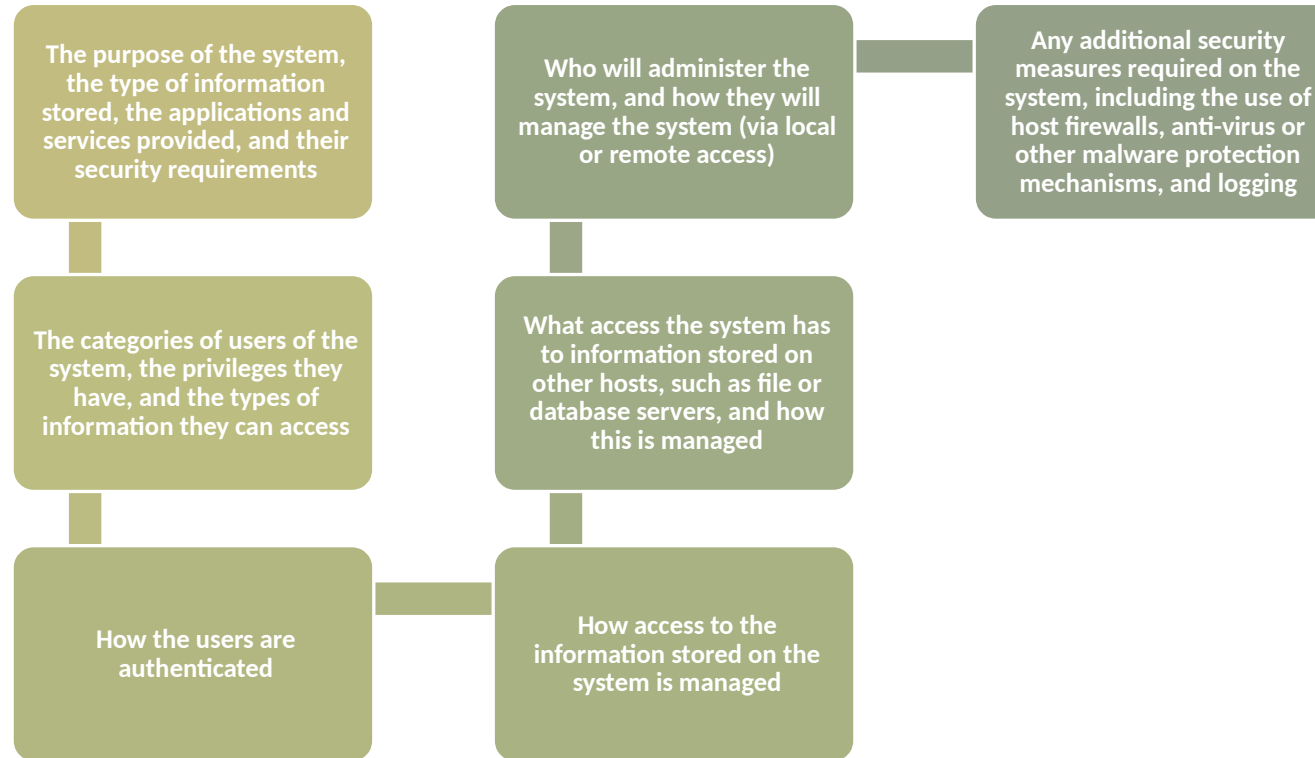
---

Possible for a system to be compromised during the installation process before it can install the latest patches

Building and deploying a system should be a planned process designed to counter this threat

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



# System Security Planning Process

# Operating Systems Hardening

---

First critical step in securing a system is to secure the base operating system

## Basic steps

- 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 system (IDS)
- Test the security of the basic operating system to ensure that the steps taken adequately address its security needs

# Security Maintenance

---

Process of maintaining security is continuous

Security maintenance includes:

- 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

# Logging

---

Can only inform you about bad things that have already happened

In the event of a system breach or failure, system administrators can more quickly identify what happened

Key is to ensure you capture the correct data and then appropriately monitor and analyze this data

Information can be generated by the system, network and applications

Range of data acquired should be determined during the system planning stage

Generates significant volumes of information and it is important that sufficient space is allocated for them

Automated analysis is preferred



### Windows systems also define privileges

- System wide and granted to user accounts

Combination of share and NTFS permissions may be used to provide additional security and granularity when accessing files on a shared resource

### User Account Control (UAC)

- Provided in Vista and later systems
- Assists with ensuring users with administrative rights only use them when required, otherwise accesses the system as a normal user

### Low Privilege Service Accounts

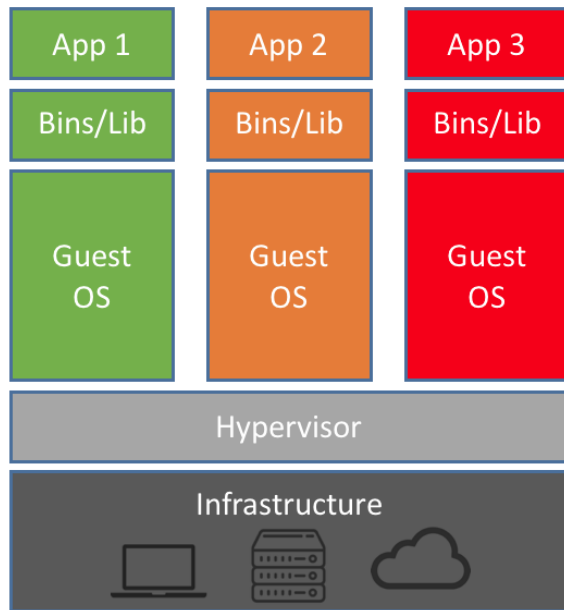
- Used for long-lived service processes such as file, print, and DNS services

# Windows Security User Administration and Access Control

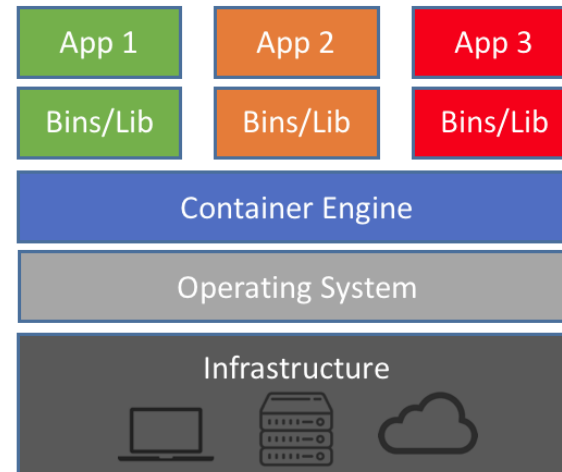
# Virtualization (VMs, Containers)

---

- ▢ A technology that provides an abstraction of the resources used by some software which runs in a simulated environment called a virtual machine (VM)
- ▢ Benefits include better efficiency in the use of the physical system resources
- ▢ Provides support for multiple distinct operating systems and associated applications on one physical system
- ▢ Raises additional security concerns



Machine Virtualization



Containers

# Containers

A recent approach to virtualization is known as container virtualization or application virtualization

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

All containerized applications on a host share a common OS kernel

For containers, only a small container engine is required as support for the containers

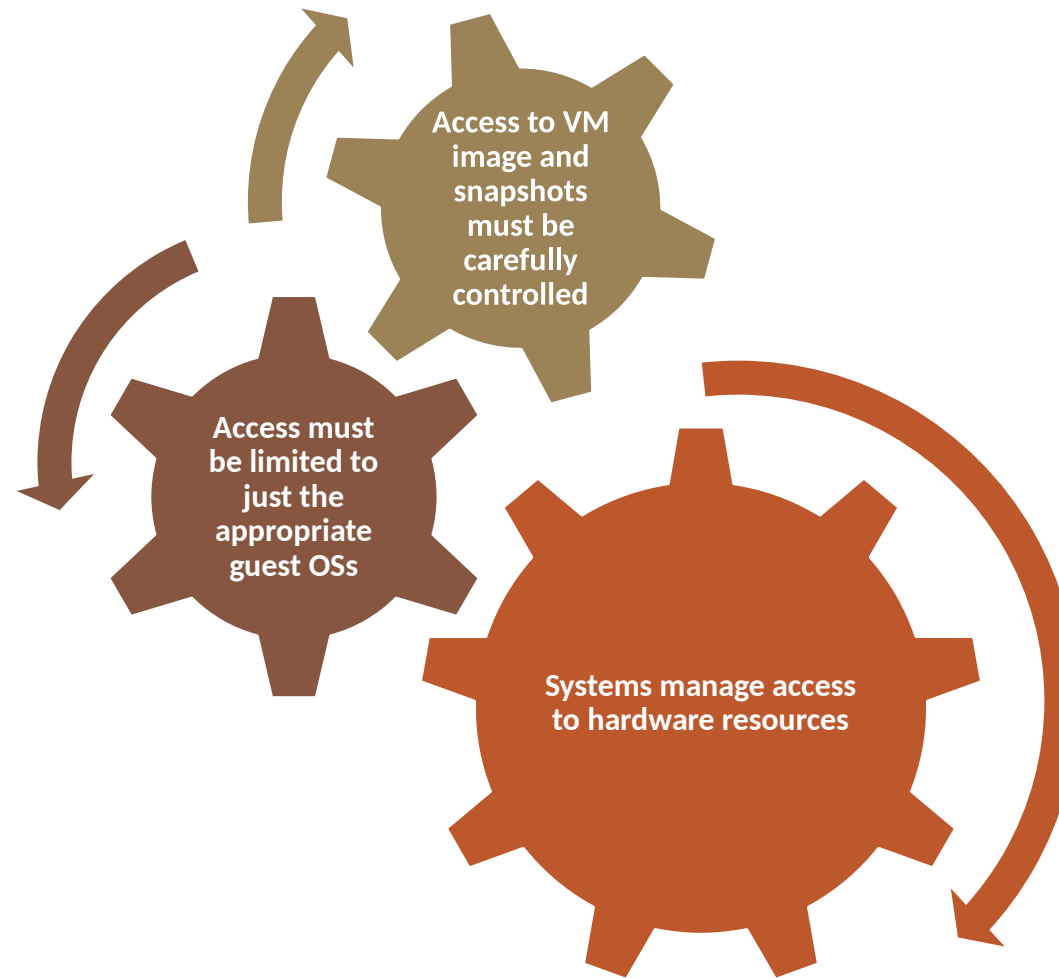
Containerization sits in between the OS and applications and incurs lower overhead, but potentially introduces greater security vulnerabilities

# Virtualization Security Issues

---

Security concerns include:

- Guest OS isolation
  - Ensuring that programs executing within a guest OS may only access and use the resources allocated to it
- Guest OS monitoring by the hypervisor
  - Which has privileged access to the programs and data in each guest OS
- Virtualized environment security
  - Particularly image and snapshot management which attackers may attempt to view or modify



# Virtualized Infrastructure Security

# Chapter 13

## Cloud and IoT Security

# Cloud Computing

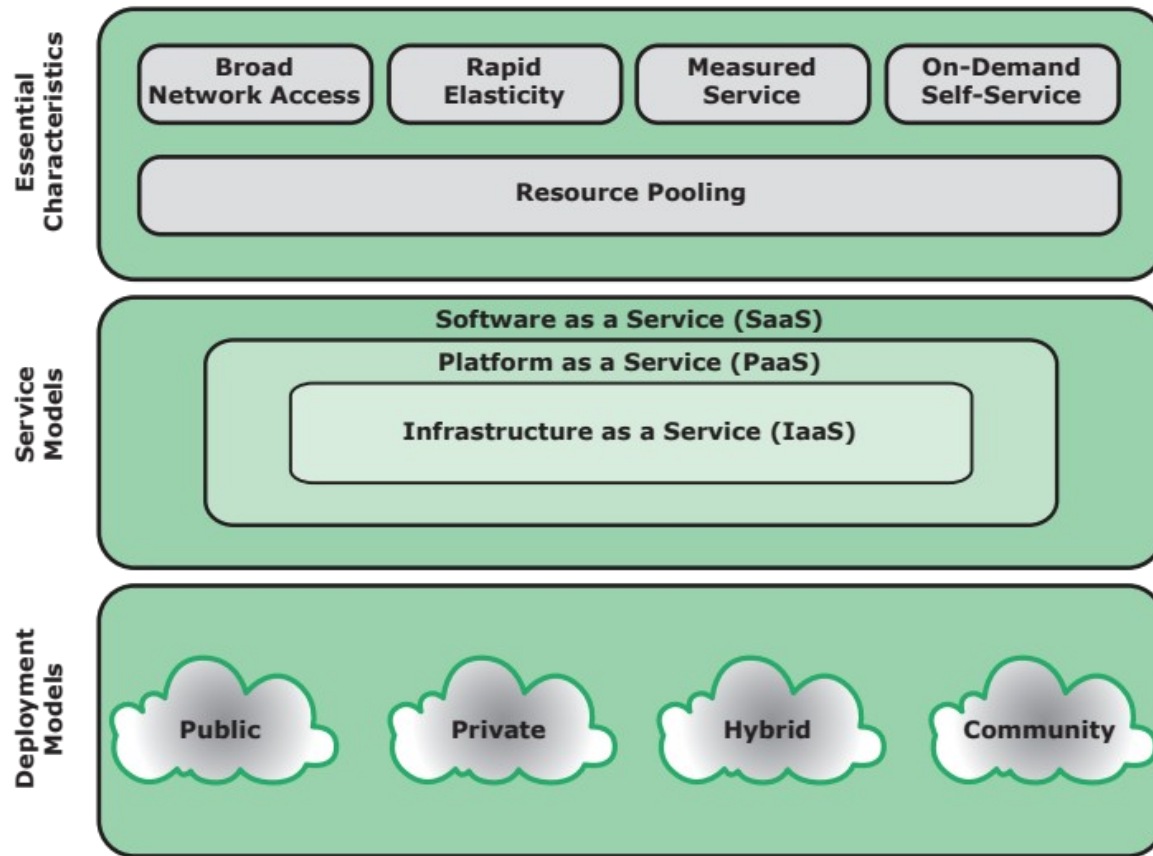
---

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.”

# Cloud Computing - Depiction

---

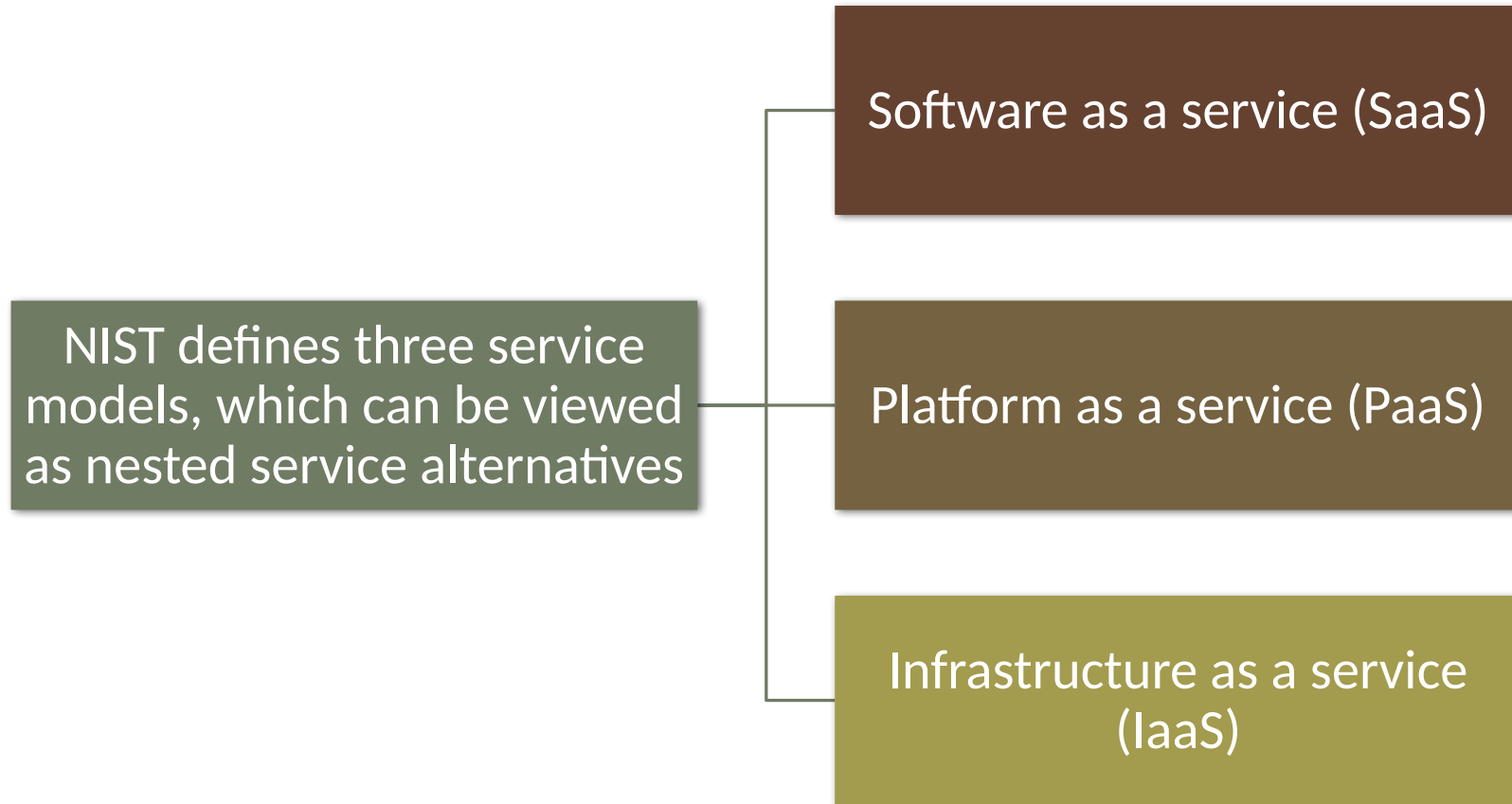


**Figure 13.1 Cloud Computing Elements**



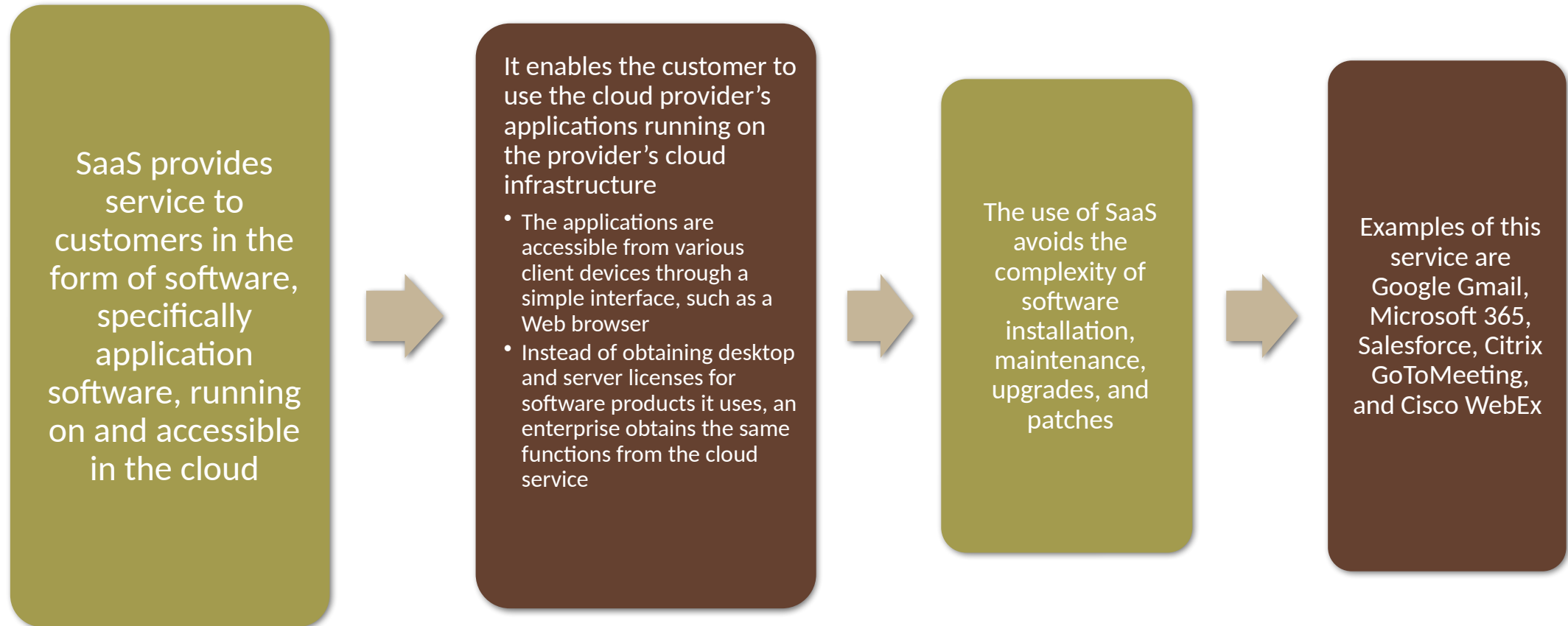
# Cloud Service Models

---



# Software as a Service (SaaS)

---



# Platform as a Service (PaaS)

---

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

It is useful for an organization that wants to develop new or tailored applications while paying for the needed computing resources only as needed, and only for as long as needed

Examples of PaaS include AppEngine, Engine Yard, Heroku, Microsoft Azure, Force.com, and Redhat Openshift

# Infrastructure as a Service (IaaS)

---

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

IaaS provides virtual machines and other virtualized hardware and operating systems

IaaS offers the customer processing, storage, networks, and other fundamental computing resources so 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

Examples of IaaS are AWS, Google Compute Engine, and IBM Cloud

# 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
  - 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, or government organization, or some combination of them
  - All major components are outside the enterprise firewall, located in a multitenant infrastructure
  - Applications and storage are made available over the Internet via secured IP, and can be free or offered at a pay-per-usage fee
- The major advantage of the public cloud is cost
- The principal concern is security

# 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

The cloud servers and storage devices may exist on premise or off premise

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 or storage as services to its branch offices

Examples of services delivered through the private cloud include database on demand, email on demand, and storage on demand

A key motivation for opting for a private cloud is security

Other benefits include easy resource sharing and rapid deployment to organizational entities

# 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

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 business

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

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

## Comparison of Cloud Deployment Models



# Security Issues for Cloud Computing

---

Security is 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

Auditability of data must be ensured

Businesses should perform due diligence on security threats both from outside and inside the cloud

- Cloud users are responsible for application-level security
- Cloud vendors are responsible for physical security and some software security
- Security for intermediate layers of the software stack is shared between users and vendors

Cloud providers must guard against theft or denial-of-service attacks by their users and users need to be protected from one another

Businesses should consider the extent to which subscribers are protected against the provider, especially in the area of inadvertent data loss

# The Internet of Things (IoT)

---

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 supports the interconnectivity 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

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

# Evolution of IoT

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

## Information technology (IT)

PCs, servers, routers, firewalls, and so on, bought as IT devices by enterprise IT people, primarily using wired connectivity

## Operational technology (OT)

Machines/appliances with embedded IT built by non-IT companies, such as medical machinery, SCADA, process control, and kiosks, bought as appliances by enterprise OT people, primarily using wired connectivity

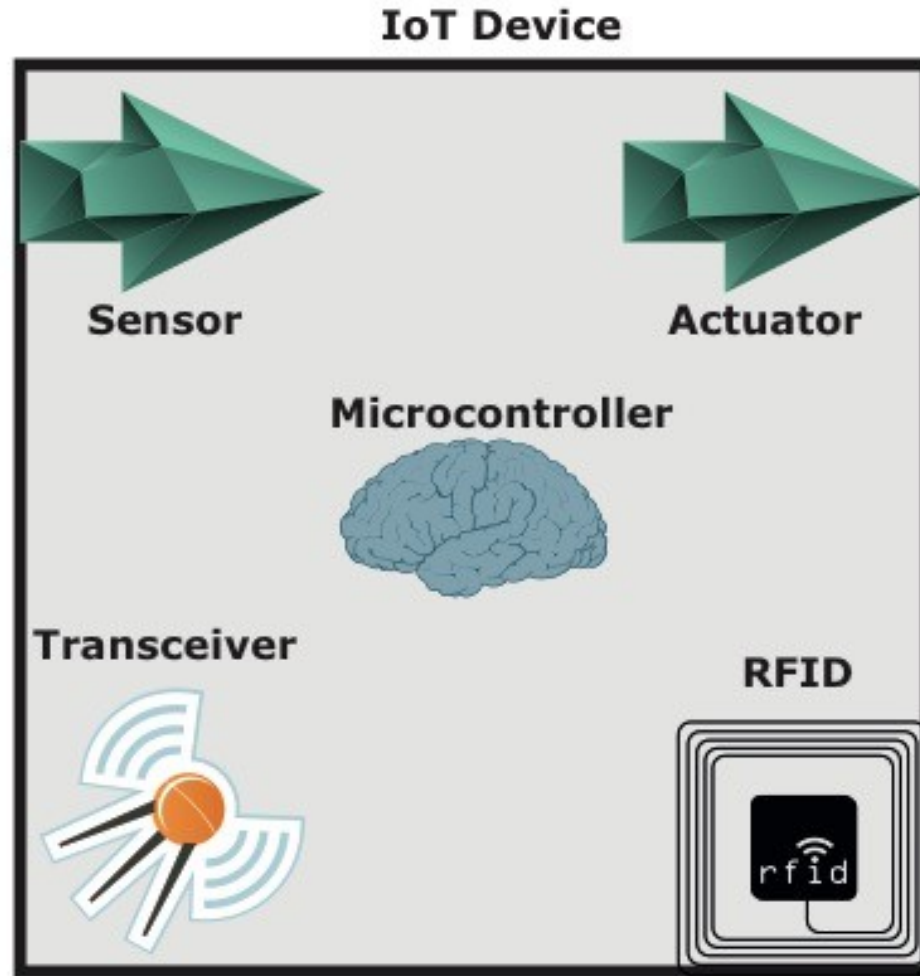
## Personal technology

Smartphones, tablets, and eBook readers bought as IT devices by consumers (employees) exclusively using wireless connectivity and often multiple forms of wireless connectivity

## Sensor/actuator technology

Single-purpose devices bought by consumers, IT and OT people exclusively using wireless connectivity, generally of a single form, as part of larger systems

It is the fourth generation that is usually thought of as the IoT, and which is marked by the use of billions of embedded devices



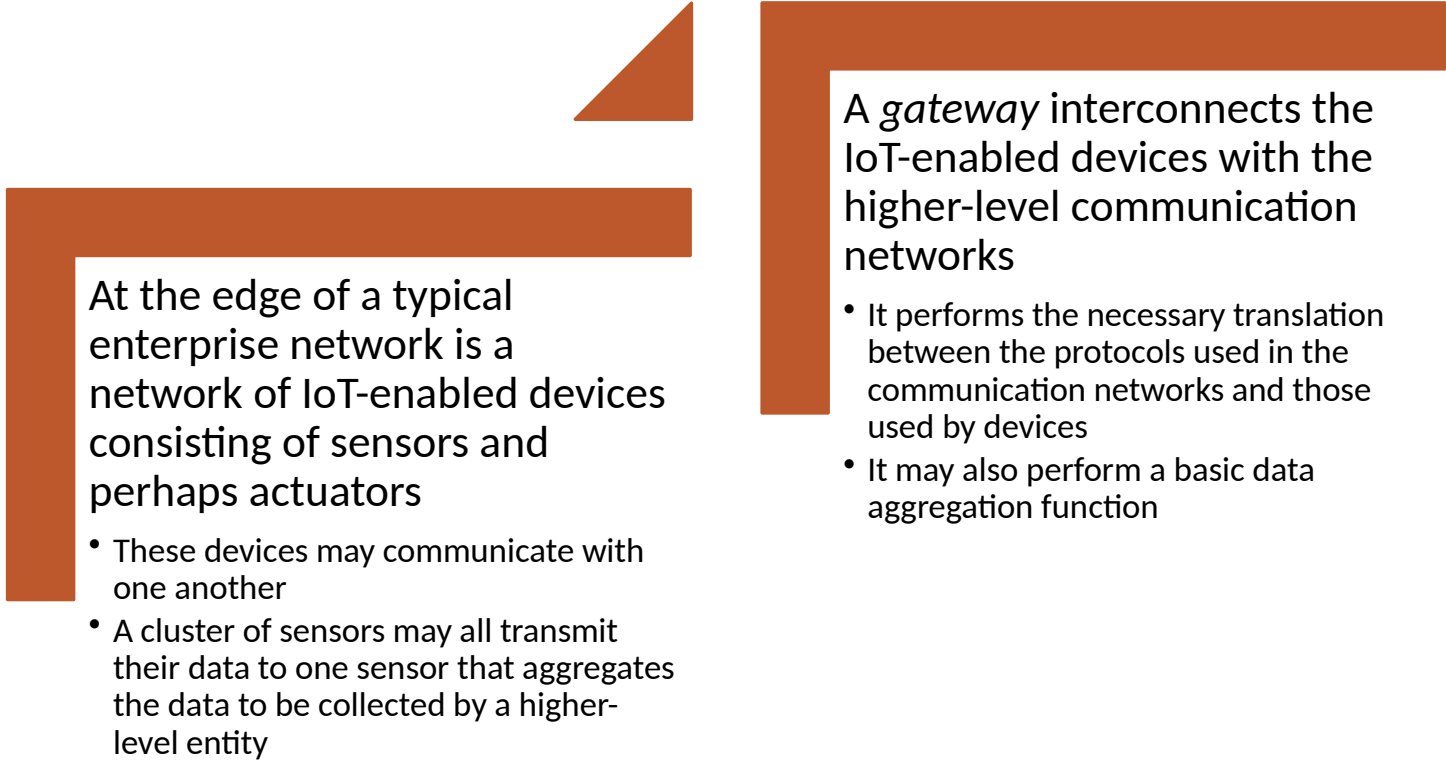
**Figure 13.8 IoT Components**

# IoT Components

---

# 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 with one another
- A cluster of sensors may all transmit their data to one sensor that aggregates the data to be collected by a higher-level entity

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
- It may also perform a basic data aggregation function

# Fog

---

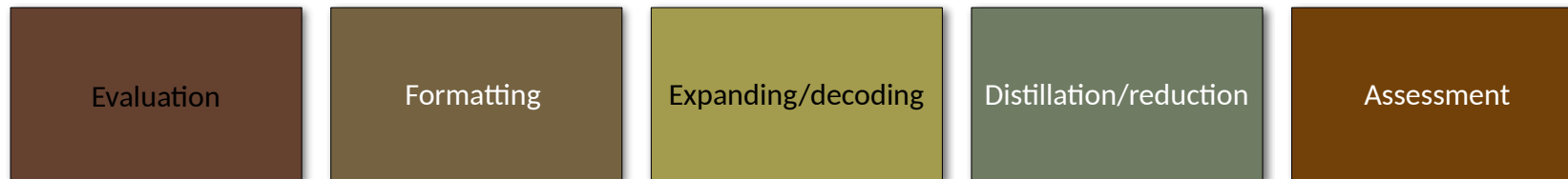
In many IoT deployments, massive amounts of data may be generated by a distributed network of sensors

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

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:



	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

# Cloud vs. Fog

---

# IoT Security and Privacy Requirements

---

ITU-T Recommendation Y.2066 includes a list of security requirements for the IoT

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

The requirements are:

- Communication security
- Data management security
- Service provision security
- Integration of security policies and techniques
- Mutual authentication and authorization
- Security audit



# IoT Gateway Security

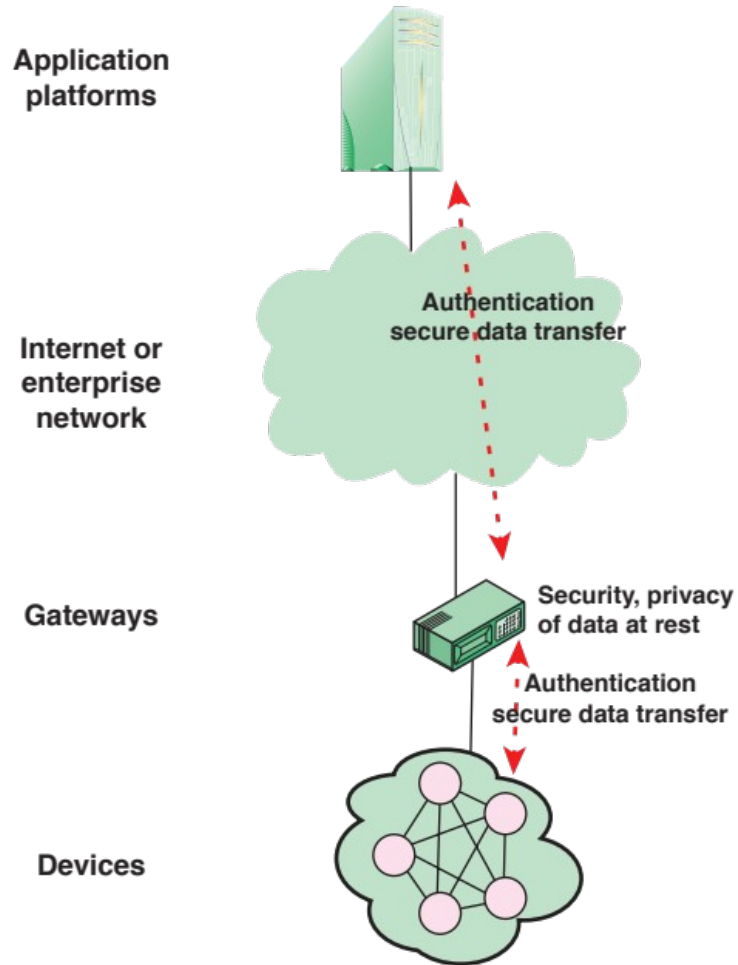


Figure 13.11 IoT Gateway Security Functions