Module-2

—

Mrs. Harinakshi C

MODULE - II

Infrastructure Security: Infrastructure Security: The Network Level, Infrastructure Security: The Host Level, Infrastructure Security: The Application Level.

Data Security and Storage: Aspects of Data Security, Data Security Mitigation, Provider Data and Its Security.

Infrastructure Security: The Network Level

- When looking at the **network level** of infrastructure security, it is important to distinguish between **public clouds and private clouds**, With private clouds, there are no new attacks, vulnerabilities, or changes in risk specific to this topology that information security personnel need to consider.

- The security considerations you have today apply to a private cloud infrastructure, too. **And the security tools you have in place (or should have in place) are also necessary for a private cloud and operate in the same way.**
- Figure shows the topological similarities between a secure extranet and a private cloud.

1. **Public vs. Private Clouds**
   - It is important to distinguish between **public clouds** and **private clouds**.
   - In **private clouds**, there are no new attacks, vulnerabilities, or changes in risk unique to the topology — so existing security practices still apply.
2. **Security Considerations**
   - The same **security measures** you already use for your current infrastructure also apply to a **private cloud**.
   - **Security tools** (e.g., firewalls, intrusion detection, monitoring) that are necessary for traditional setups are equally required for private clouds.
3. **Network Topology Similarity**
   - The **network topology** of a private cloud is similar to a **secure extranet**, meaning both require strong access controls, monitoring, and encryption to protect data.

What is an extranet?

- A secure extranet is basically a private network that allows controlled access to external partners (like suppliers, vendors, or business collaborators) using internet technologies, but with strong security controls such as authentication, encryption, and firewalls.

- An intranet = internal private network for employees.
- An extranet = extends part of that private network to trusted outsiders.
- A secure extranet = does this safely, making sure only authorized external users can access certain resources.

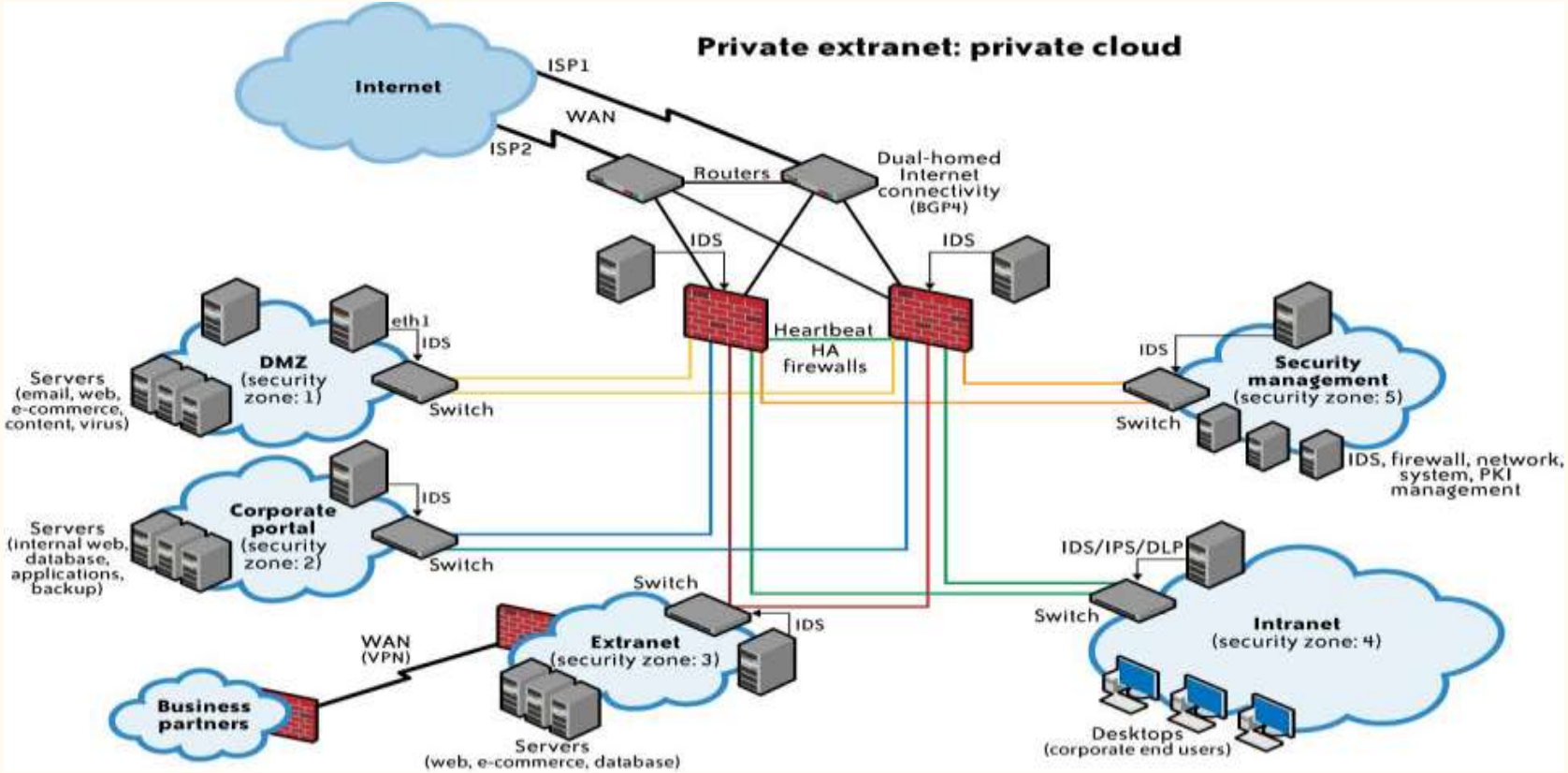
Example:

- A large enterprise (say a manufacturer) may set up a secure extranet so that its suppliers can:
  - Check raw material orders,
  - Update delivery schedules,
  - Share logistics data.
- But the suppliers **cannot** see internal company data (like HR records, finances, or confidential projects). Security tools like **VPNs, SSL encryption, and access control policies** keep it safe.

Some use cases for extranets include the following examples:

- Exchanging large volumes of data using electronic data interchange;
- Sharing product catalogs exclusively with wholesalers;
- Collaborating with other companies on joint development projects;
- Jointly developing and using training programs with other companies;
- Providing services to a group of other companies, such as an online banking application managed by one company on behalf of affiliated banks; and
- Sharing news of common interest exclusively with partner companies.

Generic network topology for private cloud computing



1. **Internet & Routers**
   - Dual-homed internet connectivity (multiple ISPs) ensures reliability.
   - Routers connect the cloud infrastructure to the internet.
2. **Firewalls & IDS (Intrusion Detection Systems)**
   - **HA firewalls** (High Availability) protect internal systems.
   - IDS monitors traffic for suspicious activity.
3. **Security Zones**
   - **DMZ (Zone 1):** Hosts public-facing servers (email, web, e-commerce).
   - **Corporate Portal (Zone 2):** Internal services (databases, applications, backup
   - **Extranet (Zone 3):** Connects **business partners** securely (via VPN).
   - **Intranet (Zone 4):** Internal users/desktops.
   - **Security Management (Zone 5):** Centralized security control (firewall, IDS, P
4. **Business Partners (Extranet Access)**
   - Access to certain servers via VPN, but **isolated** from sensitive data.
5. **End Users (Intranet Access)**
   - Employees securely access corporate apps, databases, and cloud services.

Why This Matters for Private Cloud Security

- The segmentation into zones ensures attackers can’t easily move inside the network.
- Firewalls, IDS/IPS, and DLP protect data at every layer.
- Extranet access lets trusted partners in without exposing core corporate resources.

- However, if you choose **to use public cloud services**, changing security requirements will require changes to your network topology.
- Must address how **your existing network topology interacts** with your cloud provider's network topology.
- **<u>There are four significant risk factors in this use case:</u>**

There are four significant risk factors in this use case:

1. • Ensuring the confidentiality and integrity of your organization's data-in-transit to and from your public cloud provider

2. • Ensuring proper access control (authentication, authorization, and auditing) to whatever resources you are using at your public cloud provider

3. • Ensuring the availability of the Internet-facing resources in a public cloud that are being **used by your organization**, or have been assigned **to your organization** by your public cloud providers

4. • Replacing the **established model of network zones and tiers with domains- refers to a shift in network architecture design.**

Ensuring Data **Confidentiality and Integrity**

Some resources and data previously confined to a private network are now **exposed to the Internet, and to a shared public network belonging to a third-party cloud provider.**

An example of problems associated with this first risk factor is an Amazon Web Services (AWS) security vulnerability reported in December 2008.* In a blog post, the author detailed a flaw in the digital signature algorithm used when "... making Query (aka REST) requests to Amazon SimpleDB, to Amazon Elastic Compute Cloud (EC2), or to Amazon Simple Queue Service (SQS) over HTTP." Although use of HTTPS (instead of HTTP) would have mitigated the integrity risk, users not using HTTPS (but using HTTP) did face an increased risk that their data could have been altered in transit without their knowledge.

- In **December 2008**, a **security vulnerability** was reported in **Amazon Web Services (AWS)**.
- The issue was with the **digital signature algorithm** used when sending **Query (REST) requests** to AWS services like:
  - **Amazon SimpleDB** (a database service)
  - **Amazon EC2** (a compute service)
  - **Amazon SQS** (a messaging service)
- Normally, AWS requests are digitally signed so that the server can confirm:
  - The request really came from the user.
  - The request was not altered while traveling across the internet.
- The flaw meant that, when these requests were sent using **HTTP** (which is unencrypted), there was a chance that an **attacker could intercept the request and alter it** while in transit.
- Example: If you send a request to **start a virtual machine** or **store data**, an attacker could change the request before it reached AWS.

- However, if users sent requests via HTTPS (encrypted communication), the encryption would protect the integrity of the data. This would mitigate the risk, because even if someone intercepted the data, they wouldn’t be able to read or modify it.
- The problem was that not all users enabled HTTPS, and those using HTTP were at risk.

AWS S3 Data Breaches (2017–2018)

- Multiple companies (Accenture, Dow Jones, Verizon, U.S. Army intelligence documents) exposed data due to **misconfigured Amazon S3 buckets**.
- Some buckets allowed **public HTTP access without authentication**, meaning data could be read or altered.
- Like the 2008 case, the lack of **strong authentication/encryption** created **integrity and confidentiality risks**.

Access Control in Public Cloud

- When an organization uses a **public cloud**:
- Its resources (servers, storage, databases, etc.) are **exposed to the Internet**.
- Unlike in a private data center, the company has **little or no ability to monitor network traffic** at the provider's level.
- They have **less visibility** into logs, forensics, or investigations.
- So, access control becomes harder to enforce and audit.

Example Problem: Reused (Reassigned) IP Addresses

- **IP addresses** are limited and expensive resources.
- Cloud providers (like AWS, Azure, GCP) **recycle IP addresses:** when one customer releases an IP, it's quickly reassigned to another customer.
- This practice makes sense for providers but creates **security risks** for customers.

Mitigating IP Address Reassignment Risks

Risks of Non-Aged IP Addresses:

- **Persistence in Caches**: Old IPs may remain in **DNS/ARP caches**, leading to unauthorized access.
- **Lag Time Issues**: **Delays in updating caches can result in access to outdated IPs.**

AWS Elastic IP Solution:

- **Customer Control**: AWS introduced Elastic IPs, allowing customers to manage IP assignments.

Best Practices:

- **Monitoring**: Use tools to track DNS/ARP updates.
- **Proactive Security**: Clear caches regularly and push for robust IP aging policies.

Example: 1

- An example of the problems associated with this second risk factor is the issue of reused (reassigned) IP addresses.
- Generally speaking, cloud providers do not sufficiently **“age” IP** addresses when they are no longer needed for one customer. Addresses are usually reassigned and reused by other customers as they become available.
- From a cloud provider’s perspective this makes sense. IP addresses are a finite quantity and a billable asset.
- However, from a customer’s security perspective, the persistence of IP addresses that are no longer in use can present a problem.
- A customer can’t assume that network access to its resources is terminated upon release of its IP address.
- There is necessarily a **lag time between the change of an IP address in DNS and the clearing of that address in DNS caches.**

Example: 2

- There is a similar lag time between when physical (i.e., MAC) addresses are changed in ARP tables and when old ARP addresses are cleared from cache; an old address persists in ARP caches until they are cleared.

- This means that even though addresses might have been **changed, the (now) old addresses are still available in cache, and therefore they still allow users to reach** these supposedly non-existent resources.

Ensuring the Availability of Internet-Facing Resources

- Reliance on network security has increased because an increased amount of data or an increased number of organizational personnel now depend on externally hosted devices to ensure the availability of cloud-provided resources.
- Example:
  - **BGP prefix hijacking**
  - **DNS attacks**

Prefix hijacking

- Prefix hijacking involves announcing an **autonomous system‡ address space that belongs to someone else without her permission.**
- Such announcements often occur because of a configuration mistake, but that misconfiguration may still affect the availability of your cloud-based resources.

- **What it is**: Border Gateway Protocol (BGP) is the “map of the Internet” used by ISPs to decide how to route traffic.
- Attackers can **announce fake routes** to redirect traffic through their network.
- **Impact**: Traffic can be dropped (causing downtime) or intercepted (enabling spying).
- **Real Incident**: In 2018, attackers hijacked Amazon’s Route 53 BGP prefixes. They redirected traffic to fake cryptocurrency sites and stole **$150,000 in Ethereum**.

Prefix hijacking-Example-2

- A **NANOG 2006 study** showed that several hundred BGP misconfigurations occur each month.
- The most famous case happened in **February 2008** when **Pakistan Telecom** mistakenly announced a **bogus route** for YouTube to its upstream provider **PCCW (Hong Kong)**.
- The original intent was to **block YouTube only within Pakistan** due to controversial videos.
- Instead, the route spread globally, causing **YouTube to be inaccessible worldwide for about two hours**.
- In short: **A local filtering attempt in Pakistan accidentally triggered a global BGP hijack, knocking YouTube offline worldwide for 2 hours.**

DNS ATTACK

- What is DNS?DNS is like the phonebook of the internet.
- It translates human-readable domain names (like www.google.com) into IP addresses (like 142.250.182.14) that computers use to communicate.
- Without DNS, we would have to remember long strings of numbers (IP addresses) to access websites.

Steps

1. You type www.google.com into your browser.
2. Your computer asks the **DNS resolver** (usually provided by your ISP or a public DNS like Google DNS 8.8.8.8) to find the IP address.
3. The resolver checks its cache; if not found, it queries the **DNS hierarchy**:
   - **Root server** → sends to .com TLD server.
   - **.com server** → sends to Google's authoritative DNS server.
   - **Google's DNS server** → replies with the IP (e.g., 142.250.182.14).
4. The resolver gives the IP to your browser.
5. Your browser connects to Google's server using the IP.

DNS Attacks and Cloud Computing – Risks and Vulnerabilities

1. **DNS and Cloud Computing Risks:**
2. **Common DNS Attacks in Cloud Computing:**
3. **Vulnerabilities in DNS Protocol and Implementations:**

DNS Attacks and Cloud Computing – Risks and Vulnerabilities

1. Increased External DNS Querying

- Cloud platforms rely heavily on DNS to locate and connect services (e.g., storage buckets, load balancers, APIs).
- This creates **frequent external DNS queries** that increase the chance of interception, spoofing, or manipulation.
- Attackers can target DNS traffic to redirect users to malicious endpoints.

2. Dependency on Network Security

- Cloud services often involve **distributed teams and multiple networks** (corporate, remote, third-party).
- Each network layer depends on DNS, making the overall system more vulnerable to attacks like DNS poisoning or cache hijacking.
- Misconfigured DNS can expose sensitive workloads or enable unauthorized access.

3.Split Horizon DNS Ineffectiveness

- Split horizon DNS (different answers for internal vs. external queries) is a common security practice on-premises.
- In cloud environments, **external DNS traffic often bypasses these controls**, making internal protections less effective.
- This increases the risk of data leakage, phishing redirection, or service disruption.

Common DNS Attacks in Cloud Computing

1.DNS Cache Poisoning

- Attackers inject **false DNS records** into a resolver’s cache.
- Causes users to be redirected to malicious IPs instead of legitimate ones.
- Example: Users typing www.bank.com may be redirected to a **fake phishing site**.

2.DNS Forgery (DNS Response Forging)

- Attackers send **fake DNS responses** faster than the legitimate server.
- Often targets **Name Server (NS) records**, replacing them with attacker-controlled servers.
- Impact: Large-scale redirection of domains under a compromised DNS zone.

3.DNS Spoofing (Man-in-the-Middle)

- Attackers **intercept DNS requests** in transit and modify the responses.
- Redirects users to malicious sites without altering the resolver’s cache.
- Often combined with **packet sniffing or compromised routers**.

Vulnerabilities in DNS Protocol and Implementations

1. Protocol Vulnerabilities

- DNS was designed in the early 1980s with a focus on **functionality, not security**.
- Inherent weaknesses include:
  - **Lack of authentication** → DNS can't verify if responses come from a trusted source.
  - **Susceptible to spoofing/forgery** → attackers can inject false records.
  - **Caching exploitation** → poisoned or stale records persist in cache, redirecting traffic.
  - **No confidentiality** → DNS queries/responses are usually unencrypted (exposed to eavesdropping).
- These flaws make DNS an attractive target for **cache poisoning, spoofing, and man-in-the-middle attacks**.

2. Implementation Issues

- Even when the protocol is used correctly, **poorly implemented DNS software or configurations** increase risk.
- Examples:
  - **Default configurations** with open resolvers → allow amplification attacks (DDoS).
  - **Improper access controls** → attackers can query or modify records.
  - **Unpatched DNS servers** → contain exploitable bugs (e.g., buffer overflows).
  - **Weak logging and monitoring** → delays detection of hijacking or poisoning attempts.
- Real-world failures often come from **misconfigurations or negligence**, not just protocol flaws.

Mitigation and Prevention Strategies for DNS Attacks in Cloud Computing

1. Notable DNS Vulnerabilities:

- **Kaminsky Bug (CVE-2008-1447):**
  - **Vulnerability in DNS insufficient socket entropy**, leading to cache poisoning attacks.
  - Highlighted the ease of tricking DNS servers into accepting bad information.

- **Persistent DNS Cache Poisoning:**
  - Despite earlier fixes, DNS cache poisoning remains a significant threat in cloud environments.

- **2. Key Mitigation Strategies:**

1. **DNSSEC (Domain Name System Security Extensions)**
   - Adds **digital signatures** to DNS records.
   - Ensures **authenticity** (records come from the right source) and **integrity** (records not altered in transit).
   - Protects against **DNS cache poisoning and forgery**.
   - Example: Even if an attacker injects a fake IP for `www.bank.com`, DNSSEC validation will reject it.
2. **Split DNS Configuration**
   - Separates **internal vs. external DNS resolution**.
   - Internal DNS servers handle sensitive/private queries (e.g., company intranet, cloud workloads).
   - Reduces **external exposure** and minimizes the attack surface.
   - Prevents attackers from learning about internal systems through external queries.

3. **Regular DNS Software Updates**
   - Patch DNS servers (e.g., BIND, Microsoft DNS, Unbound, PowerDNS) to fix known vulnerabilities.
   - Prevents exploits such as buffer overflows, remote code execution, or amplification abuse.
   - Replace/disable outdated DNS services or misconfigured **open resolvers**.
4. **Monitoring and Logging**
   - Enable **DNS query logging** to detect anomalies (e.g., spikes in traffic, suspicious domains).
   - Monitor for **exfiltration attempts** (DNS tunneling often used for data theft).
   - Use security tools/IDS to flag **suspicious response patterns** or DNS hijack attempts.
   - Real-time alerts help respond before damage spreads.

3. Role of Cloud Service Providers (CSPs) in DNS Security

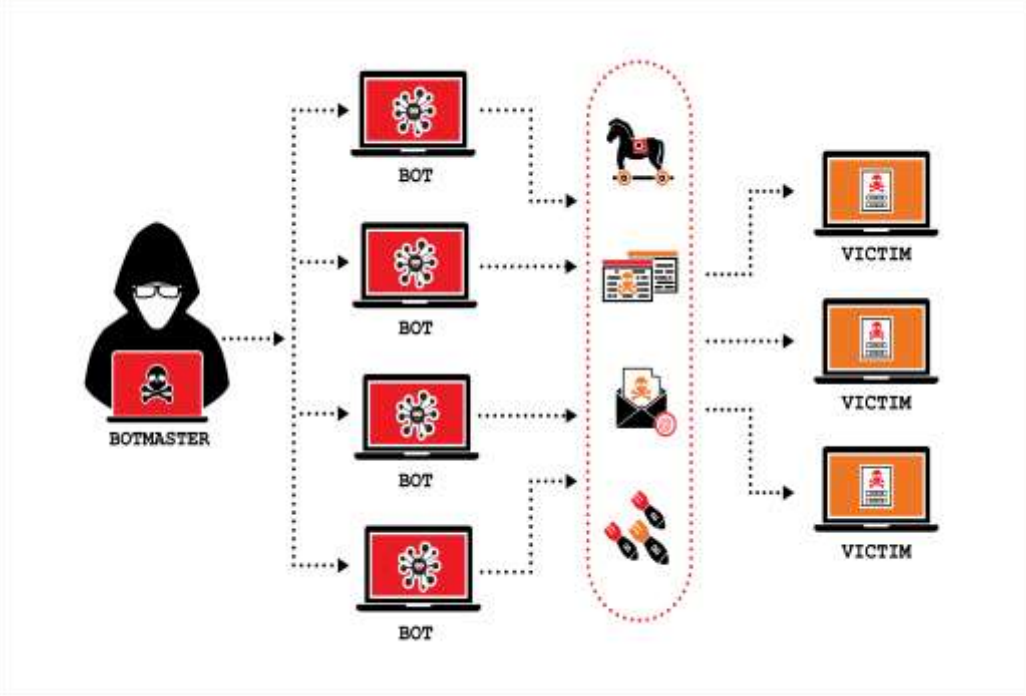
1. **DNS-Level Protection Services**
   - CSPs (AWS, Azure, GCP, Cloudflare, etc.) provide **managed DNS services** with built-in protections.
   - Examples:
     - **Automatic DNSSEC support** → ensures authenticity and integrity of DNS records.
     - **DDoS Protection** at DNS layer → absorbs large-scale attacks targeting DNS availability.
     - **Anycast Routing** → spreads DNS traffic across multiple global nodes, improving resilience.
   - These features are critical for maintaining **service availability** during DNS-based attacks.
2. **Integration with Cloud Security Ecosystem**
   - CSP DNS services often integrate with **Web Application Firewalls (WAFs)**, **Load Balancers**, and **CDNs**.
   - Helps prevent **malicious traffic redirection** and improves **fault tolerance** for applications.
   - Example: AWS Route 53 integrates with AWS Shield for DNS-based DDoS mitigation.

Distributed Denial of Service (DDoS)

- A **Distributed Denial of Service (DDoS)** attack is a type of cyberattack in which multiple compromised devices, often spread across different locations, are used to overwhelm a target (such as a server, website, or network) with a massive flood of internet traffic.

- The goal of a DDoS attack is to make the target's services unavailable to its legitimate users by **exhausting its resources, such as bandwidth, processing power, or memory.**

- DDoS = Distributed Denial of Service
- Attackers use multiple compromised systems (botnets, IoT devices, malware-infected PCs) to flood a target with traffic.
- Goal: Overwhelm the target's resources (bandwidth, CPU, memory, application capacity), making services slow or completely unavailable to legitimate users.



Key Aspects of DDoS Attacks

1. **Distributed Nature**: DDoS attacks involve multiple systems, often part of a botnet (a network of infected computers), which makes it harder to stop the attack by blocking a single source of traffic.
2. **Flooding the Target**: The attackers flood the target system with fake requests, preventing it from processing legitimate requests.

- **Impact**:
  - **Service downtime and unavailability.**
  - **Loss of revenue for businesses due to interrupted operations.**
  - **Damage to reputation if the attack is public and sustained.**

Types of DDoS Attacks:

1. **Volume-Based Attacks**
   - Flood the target with massive traffic (measured in Gbps/Tbps).
   - Examples: UDP Floods, ICMP Floods.
   - Cloud risk: Can saturate Internet links, blocking all access.
2. **Protocol Attacks**
   - Exploit weaknesses in protocols (Layer 3/4) like TCP, UDP, ICMP.
   - Examples: SYN Flood, Ping of Death, Smurf Attack.
   - Cloud risk: Exhausts firewalls, load balancers, or routers.
3. **Application Layer Attacks**
   - Target specific apps (Layer 7) like web servers, DNS, or APIs.
   - Examples: HTTP GET/POST Floods, DNS Query Floods.
   - Cloud risk: Harder to detect because traffic **looks like normal user requests**.

Mitigating DDoS Attacks:

1. CSP Protections (Cloud Service Provider Protections)

- Examples: **AWS Shield, Azure DDoS Protection, Google Cloud Armor**.
- These are **built-in protections by cloud providers** that automatically detect unusual traffic spikes.
- They analyze patterns in traffic and block malicious floods before they reach your application.
  Example: AWS Shield detects a sudden 1 million requests/second to your EC2 instance and automatically filters them before they overload your server.

2. CDN & Anycast

- **CDN (Content Delivery Network):** Distributes your website content across multiple global servers. Users connect to the nearest server instead of one central server.
- **Anycast:** A networking method where a single IP address routes users to the **closest or healthiest server** in a global network.
  Example: If your website uses **Cloudflare or AWS CloudFront**, traffic from India hits the nearest CDN server in India, while US users connect to US servers. This distributes load and makes DDoS harder.

3. Auto-Scaling

- Cloud automatically adds more servers/resources when there's a traffic spike.
- It helps absorb sudden increases in load, whether from legitimate users or attackers.
- Limitation: It can't protect against **very large DDoS attacks** forever (you'll hit cost or service limits).

  Example: If your web app usually runs on 2 servers, but traffic jumps, Azure automatically scales it to 10 servers to handle requests.

4. Rate Limiting & WAFs

- **Rate Limiting:** Restricts how many requests a user/IP can make in a certain time. Prevents flooding.
- **WAF (Web Application Firewall):** Filters HTTP(S) traffic, blocking suspicious requests (like SQL injection, bots, or floods).

  Example: A user tries to send 1000 login attempts per second. Rate limiting blocks them after 10 requests. A WAF may also detect and block attack patterns like malicious scripts.

5. Traffic Scrubbing Centers

- Specialized cloud security services where suspicious traffic is **redirected for cleaning (scrubbing)** before it reaches your app.
- They separate **good traffic** (real users) from **bad traffic** (bots/attackers). Example: Akamai or Cloudflare scrubbing center filters out a botnet DDoS attack so only valid requests reach your application servers.

Key Points on Internal DDoS Attacks in IaaS:

1. **External DDoS Attacks:**

- Suppose a company hosts its **e-commerce website** on AWS EC2.
- An attacker from outside (internet) launches a **botnet attack** using thousands of compromised devices to flood the website's public IP with traffic.
- Result: The website becomes **slow or unavailable** for legitimate customers.
- This is **External DDoS** because the attack originates from **outside the cloud provider's infrastructure** and targets a **public-facing service**.

Internal DDoS Attack Example

1. Internal Attack

- Consider multiple customers sharing the same IaaS cloud infrastructure (multi-tenancy).
- A **rogue customer** rents a VM in Azure.
- Instead of using it legitimately, they run a malicious program that floods **another customer's VM** or the **cloud provider's internal services (like storage or DNS inside Azure)**.
- Since the attack is happening **inside the provider's private/internal network**, traditional internet firewalls may not detect it.
- Result: Other tenants (customers) suffer degraded performance or outages.
- This is **Internal DDoS** because the attacker is already **inside the provider's environment** and exploits the **shared internal network**.

Key Difference

- **External** → Attack from internet → Targets **public services.**
- **Internal** → Attack from inside cloud → Targets **other tenants or provider infrastructure.**

2. Risks in IaaS Networks:

- **Shared Resources:** In an IaaS environment, customers share portions of the network that handle both public-facing and internal operations (e.g., managing Amazon Machine Images (AMIs), access to non-public instances). This creates an opportunity for internal attacks.
- **Non-Routable Networks:** These are private, internal networks not visible from the public internet but critical for managing cloud infrastructure. A malicious customer could launch a DDoS attack from within, targeting other tenants or the provider's systems.

3. Potential Attack Vectors:

- **Overloading Internal Networks**: By consuming excessive bandwidth or generating too many requests, an attacker could disrupt communication between the IaaS provider and other customers.
- **Targeting Non-Public Instances**: Since customers often use non-public instances for sensitive workloads, attackers might attempt to overload these systems through internal DDoS mechanisms.
- **Compromising Provider Infrastructure**: A rogue tenant could attack the IaaS provider's management systems (such as those controlling physical servers or network management) to degrade performance or cause outages.

4. Lack of Detection and Prevention:

- **Detective Controls**: The IaaS provider might lack adequate monitoring for internal attacks, especially if the attack traffic is not routed through the internet.
- **Preventive Controls for Customers**: The only defense for customers in many cases is the **hardening** of their instances, ensuring that internal services and access points are secured.

Preventive Strategies for Internal DDoS in IaaS:

1. Instance Hardening:
    1. Secure non-public instances (such as AMIs) by applying strict firewall rules and access controls.
    2. Regularly update and patch software to prevent known vulnerabilities from being exploited.
2. Segmentation and Firewalls:
    1. Use network segmentation to isolate sensitive workloads from the rest of the internal network.
    2. Implement firewalls or security groups (e.g., AWS Security Groups) to control traffic flow between instances.

3. Monitoring and Intrusion Detection:

- Use advanced monitoring tools or third-party services to detect unusual traffic patterns or resource consumption within the internal network.

- Implement intrusion detection systems (IDS) to monitor network activity for signs of malicious behavior

4. Collaboration with Providers:

- Work with the IaaS provider to ensure they are monitoring their internal networks for potential attacks and have systems in place to alert customers of any suspicious activity.

- Request additional security services, such as **DDoS protection** or managed firewall services, that can help mitigate attacks.

Replacing the Established Model of Network Zones and Tiers with Domains

- **Traditional Model: Network Zones and Tiers**
- Earlier, networks were divided into **zones** such as:
  - **External (Internet)** – completely untrusted.
  - **DMZ (Demilitarized Zone)** – semi-trusted area where public-facing apps (e.g., web servers) are hosted.
  - **Internal Network** – highly trusted area (databases, internal apps, employees).

- Security was enforced by firewalls and access control lists between these **zones.** This model assumes a **clear inside vs outside boundary**.

Problem with the Old Model

- In cloud and modern distributed systems, boundaries are **blurry**:
  - Users connect from anywhere (not just corporate LAN).
  - Services may run across multiple cloud providers.

- Meaning → the concept of "inside is trusted, outside is untrusted" no longer works.

New Model: Domains

- Instead of broad zones, we move towards **domains of trust**.
- A **domain** is defined by **identity, context, and security policy**, not just network location.
- Example domains:
  - **User Domain** → defines trust level of users based on MFA, device posture, etc.
  - **Application Domain** → microservices and apps controlled by IAM + Zero Trust.
  - **Data Domain** → access to sensitive data controlled by encryption + RBAC.
- This is aligned with the **Zero Trust Model**, where *no implicit trust is given based on network location*.

  Each access request must be **authenticated and authorized individually**.

Example

- In old model:
  - If a device is inside corporate LAN, it's trusted.
  - Attackers who compromise one internal machine can move laterally easily.
- In new model (domains):
  - Even if inside the network, each request from that device must prove identity, meet security conditions, and be authorized.
  - Example: A developer's laptop inside office still needs MFA + device compliance to access the source code repo.

Private Cloud Advantages:

1. Reduced Risks:

- A **private cloud**, particularly one internal to your organization’s network, reduces many risks compared to public clouds. Since resources are dedicated and isolated, potential security threats from other tenants are minimized.
- **Private clouds hosted at a cloud provider's facility** still reduce risks but to a lesser extent, depending on the provider's security capabilities and maturity.

2. Encryption and Confidentiality:

- **Encryption for Data-in-Transit**: Encrypting data as it moves between systems is crucial for maintaining confidentiality. Using **validated cryptographic implementations** (like AES, TLS) ensures that data remains secure against interception and unauthorized access.
- **Digital Signatures**: Digital signatures ensure the integrity of data by making it nearly impossible to tamper with. This provides verifiable proof that the data has not been altered.

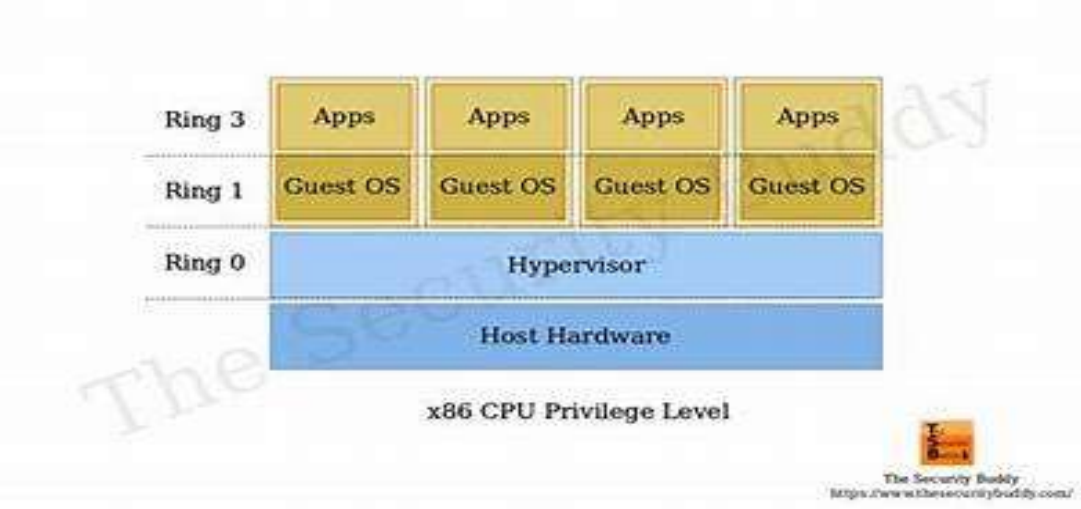
Infrastructure Security: The Host Level

- Cloud Delivery Model
    - Consider security risks across SaaS, PaaS, and IaaS delivery models, and deployment models (public, private, hybrid).
- Virtualization Security Threats
    - Existing threats like VM escape, system configuration drift, and insider threats (via weak access control to hypervisors) extend into cloud environments.
- Elasticity of Cloud Computing:
    - The dynamic nature of cloud environments creates challenges in security management due to rapid provisioning and short-lived VMs.

- Vulnerability Management:
  - Managing vulnerabilities and patches is more complex than in traditional data centers due to the high rate of change.
- Velocity of Attack:
  - Cloud environments with many compute nodes and homogeneous operating systems can amplify security threats quickly.
- Trust Boundaries:
  - Distinguish between your security responsibilities as a host manager and the cloud service provider's (CSP) responsibilities in managing host infrastructure security.

VM escape

- A VM escape occurs when an attacker exploits a vulnerability in the virtualization software or guest operating system to gain control over the hypervisor or host system.
- By doing so, they can penetrate the entire virtualized environment and potentially compromise all other virtual machines running on the same host.



System configuration drift

- Configuration drift is when configurations in an IT system gradually change over time.
- Drift is often unintentional and happens when undocumented or unapproved changes are made to software, hardware, and operating systems.
- It can have an impact on system performance and security.

The dynamic nature (elasticity)

- **Elasticity in Cloud**
  Cloud computing allows resources (like Virtual Machines, storage, etc.) to scale up or down very quickly depending on demand.
- **Security Challenge**
  Because resources are created and destroyed rapidly, traditional security methods (like manual patching, slow vulnerability scanning) don't keep up.
  - **Rapid provisioning**: New VMs appear/disappear in seconds.
  - **Difficult patch management**: Vulnerabilities can spread before patches are applied.
  - **High rate of change**: Much faster than in a traditional data center.
- **Key Point** → Elasticity improves efficiency but makes security management harder.

Velocity of Attack in the Cloud

•**Amplified Threats**

Clouds use **thousands of compute nodes** with similar operating systems.

• This uniformity means a single vulnerability can spread very fast.

→ This is called the **“velocity of attack”** in cloud environments.

•**Shared Responsibility**

•Customers (you) must secure the **part of the infrastructure you manage**

- (e.g., applications, user access, configurations).

•Cloud Service Providers (CSPs) secure the **underlying infrastructure**

- (e.g., hardware, hypervisor, data center security).

•**Remember :** You must clearly **understand the trust boundary**

•—where your responsibility ends and the CSP’s responsibility begins.

•Cloud elasticity = **speed + flexibility**, but also **rapid attack surface changes**.
•Security must adapt to this fast-paced environment with automation,
• continuous monitoring, and shared responsibility.

SaaS and PaaS Host Security

- **Limited Information Disclosure**

  CSPs don't reveal much about their **host platforms, operating systems, or internal security processes**.

  → Reason: If hackers knew these details, they could exploit them to attack the service.

- **Opaque to Customers**

  In **SaaS (Software-as-a-Service)** like Salesforce, Workday, or **PaaS (Platform-as-a-Service)** like Google App Engine or Force.com:

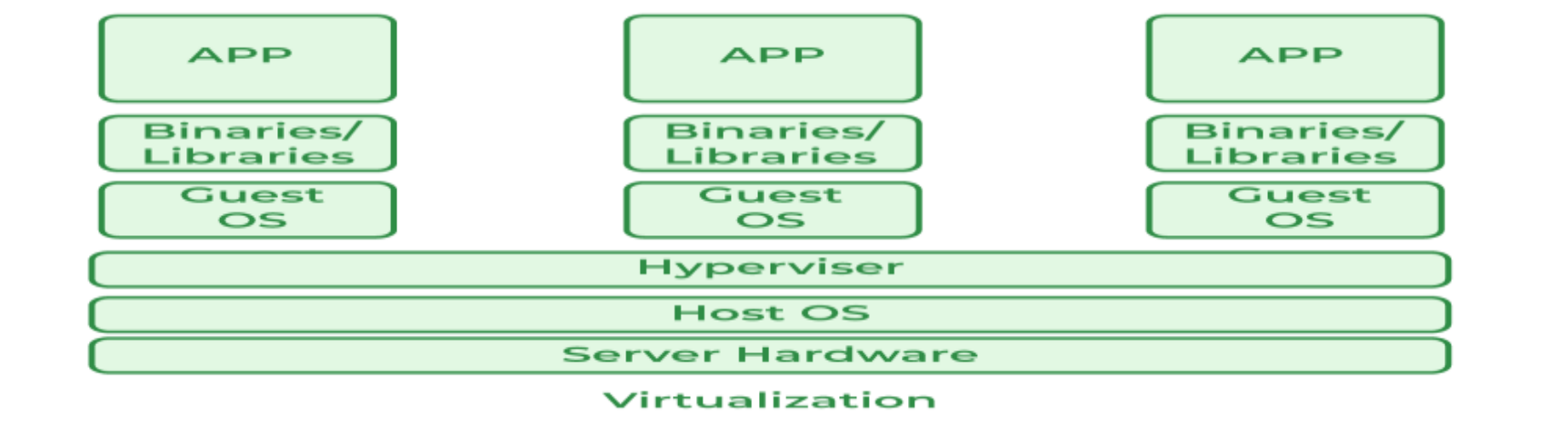
  - Customers don't have visibility into how the hosts are secured.
  - **Responsibility for host security lies entirely with the CSP.**

- **Assurance and Virtualization**

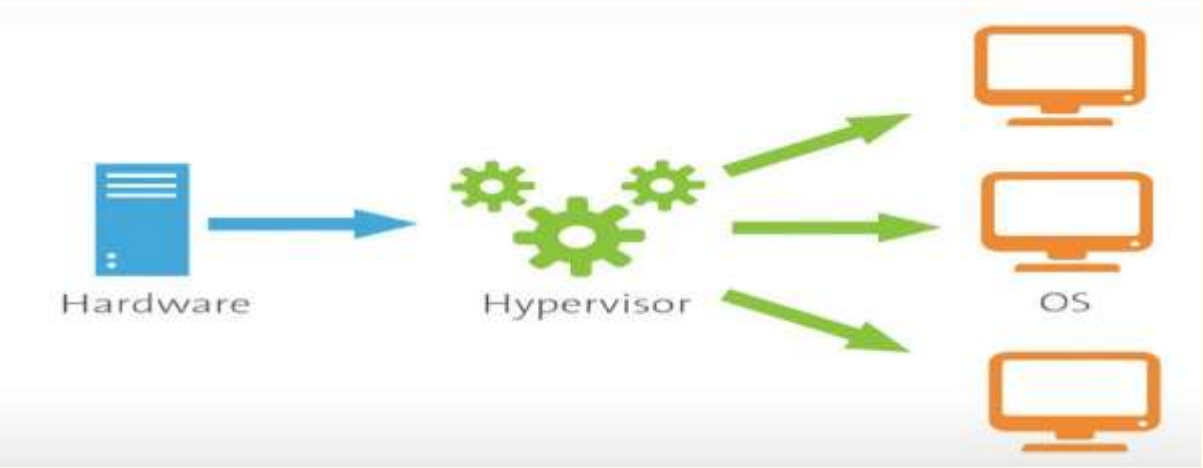
  - How to Get Assurance from CSPs
    - Customers can request security details through an **NDA** (nondisclosure agreement).
    - Or demand assurance through **standards/frameworks** like **SysTrust, ISO 27002**.
  - Control Assurance
    - CSPs must prove that they have **preventive controls (to stop attacks)** and **detective controls (to identify breaches quickly)**.
    - This is usually validated by **third-party audits or ISO certification**.
  - Virtualization in Host Security
    - CSPs rely on **virtualization platforms** (e.g., Xen, VMware) to maximize hardware utilization and isolation between tenants.
    - Virtualization is a **key enabler** for scalability and secure multi-tenant environments.

Virtualization

- Virtualization is a process that allows a computer to share its hardware resources with multiple digitally separated environments.
- Each virtualized environment runs within its allocated resources, such as memory, processing power, and storage.
- With virtualization, organizations can switch between different operating systems on the same server without rebooting.



- A hypervisor is a software that you can use to run multiple virtual machines on a single physical machine.
- Every virtual machine has its **own operating system and applications.**
- The hypervisor allocates the underlying physical computing resources such as **CPU and memory to individual virtual machines as required.**



How does a hypervisor work?

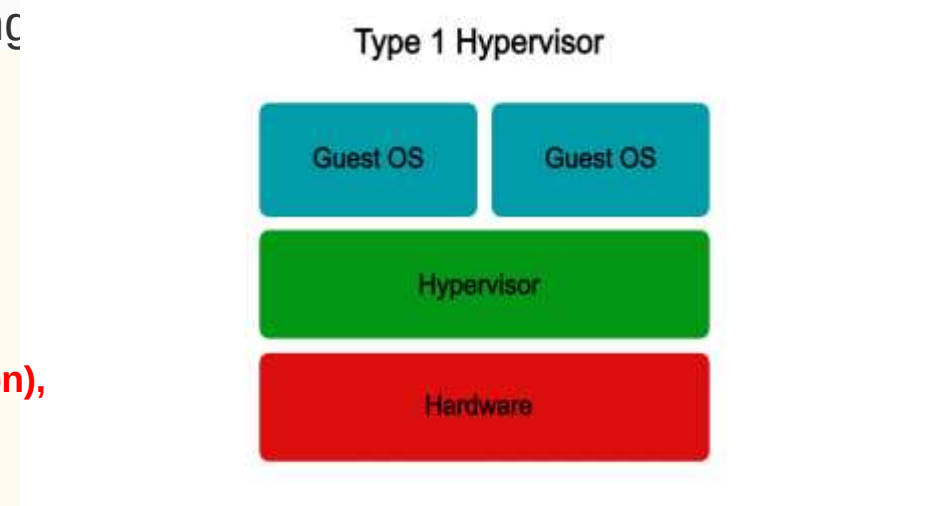
- **System administrators** install the hypervisor software on physical servers.
- The hypervisor loads the virtual machine images to create multiple virtual operating systems.
- The physical machine **is known as a *host*,** and the virtual operating systems are ***guests*.**

Resource allocation

- The hypervisor ensures that **each virtual machine receives the allocated resources as** configured. It does so by acting as an intermediary between guest machines and the underlying physical hardware.
- The hypervisor relays requests for processing power, memory, storage, and other resources to the host machine in several ways, including API calls.
- An **API is a software communication method that allows different applications** to exchange data.

Type 1 hypervisor

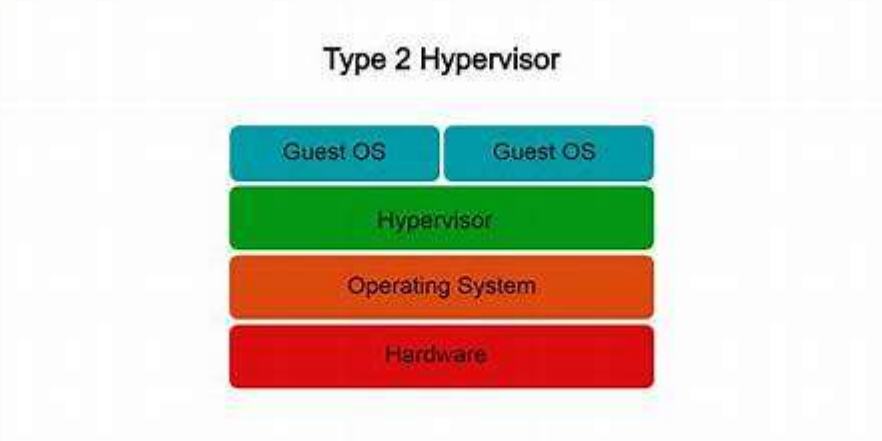
- The type 1 hypervisor sits on top of the metal server and has direct access to the hardware resources. Because of this, the type 1 hypervisor is also known as a bare-metal hypervisor. The host machine does not have an operating system installed in a bare-metal hypervisor setup. Instead, the hypervisor software acts as a lightweight operating



VMware ESXi, Microsoft Hyper-V (bare-metal version), KVM, Citrix XenServer.

Type 2 hypervisor

- The type 2 hypervisor is a hypervisor program installed on a host operating system. It is also known as a hosted or embedded hypervisor. Like other software applications, **hosted hypervisors do not have complete control of the computer resources. Instead, the system administrator allocates the resources for the hosted hypervisor, which it distributes to the virtual machines**.



- When dealing with **cloud providers**, it’s important to understand **how they implement virtualization** and what **security controls** they apply to protect the virtualization layer.

- This is because vulnerabilities at the virtualization layer could potentially allow attackers to bypass isolation and gain unauthorized access to other tenants’ data or resources.

- Both **PaaS (Platform as a Service)** and **SaaS (Software as a Service)** platforms use a **host abstraction layer** to hide the underlying host operating system from end users.
- **SaaS**:
  - End users only see the final application (e.g., Gmail, Salesforce).
  - The **host abstraction layer is completely invisible** to them.
  - Only **developers and the CSP's operations team** can interact with it for maintenance, patching, and scaling.

PaaS:

- Users (developers) are given **indirect access** to the host abstraction layer.
- This access comes via **platform APIs** (e.g., database services, runtime environments, messaging queues).
- These APIs internally communicate with the **host abstraction layer, allowing developers to build, deploy, and manage applications without dealing directly with the OS or hardware.**

- In short, if you are a SaaS or a PaaS customer, you are relying on he CSP to provide a secure host platform on which the SaaS or PaaS application is developed and deployed by the CSP and you, respectively.

What is a Tenant?

- A **tenant** is basically a **customer (organization or individual)** that is using a **shared cloud service**.

- In **multi-tenant environments**, multiple tenants share the same **infrastructure** (servers, storage, networking, virtualization layer), but each tenant's **data and applications are logically isolated** from the others.

- **Example in Different Cloud Models:**
- **SaaS**
  - Many companies may use **the same SaaS platform** (e.g., Microsoft 365, Salesforce).
  - Each company is a **tenant**.
  - They share the same underlying infrastructure, but their data is isolated.
- **PaaS**
  - Multiple developers or organizations use the same **platform services** (e.g., Google App Engine, AWS Elastic Beanstalk).
  - Each developer group is a **tenant**, running apps on the same hardware and virtualization layer.
- **IaaS**
  - Customers renting VMs or storage on the same physical servers are different **tenants**.
  - The hypervisor ensures each tenant’s VM is isolated from others.

Virtualization-Cloud

- Virtualization uses specialized software, called a hypervisor, to create several cloud instances or virtual machines on one physical computer.
- **Cloud instances or virtual machines**
- After you install virtualization software on your computer, you can create one or more virtual machines. You can access the virtual machines in the same way that you access other applications on your computer. Your computer is called the host, and the virtual machine is called the guest. Several guests can run on the host. Each guest has its own operating system, which can be the same or different from the host operating system.

IaaS Host Security

- IaaS customers are primarily responsible for securing the hosts provisioned in the cloud.
- Given that almost all IaaS services available today employ virtualization at the host layer, host security in IaaS should be categorized as follows:
  - *Virtualization software security*
  - *Customer guest OS or virtual server security*

Virtualization Software Security

- Since the CSP manages the virtualization software that sits on top of the hardware, **customers will have neither visibility nor access to this software.**

- Hardware or OS virtualization enables the sharing of hardware resources across multiple guest VMs without interfering with each other so that you can safely run several operating systems and applications at the same time on a single computer.

- IaaS services are using “bare metal hypervisor” technologies (also known as type 1 hypervisors), such as VMware ESX, Xen, Oracle VM, and Microsoft’s Hyper-V. These hypervisors support a variety **of guest OSs, including Microsoft Windows, various Linux “flavors,” and Sun’s OpenSolaris.**

Virtualization Software Security

- **No Customer Access**
  - The **CSP manages the virtualization software** (the hypervisor) that runs on the hardware.
  - Customers have **no visibility or control** over this layer.
- **Why Virtualization Matters**
  - Hardware/OS virtualization lets multiple VMs run **on the same hardware** safely, without interfering with each other.
  - This enables different **operating systems and applications** to coexist on one machine.
- **Bare Metal Hypervisors in IaaS**
  - IaaS providers use **Type-1 hypervisors** (bare metal), such as **VMware ESX, Xen, Oracle VM, Microsoft Hyper-V**.
  - These support a variety of **guest OSs** like Windows, Linux variants, and OpenSolaris.
- **Key Point** → Virtualization is the backbone of IaaS, but customers must rely fully on the CSP for its security.

Why Hypervisor Security is Critical

- **Isolation & Multi-tenancy**
  - Hypervisors isolate customer VMs from one another.
  - Any weakness could let one tenant **break into another tenant's VM**.
- **New Security Arms Race**
  - Hackers vs CSPs are in a continuous battle over hypervisor vulnerabilities.
- **High-Risk Target**
  - If virtualization is compromised, **all VMs on that infrastructure** could be exposed.
  - This could be **catastrophic** to every customer using that cloud.
- **Key Point** → The hypervisor is the "crown jewel" for attackers; protecting it is absolutely critical.

Actions (Mitigation)

- **CSP Responsibilities**
  - Enforce strong controls: restrict physical/logical access to hypervisors.
  - Apply strict policies and monitoring across virtualization layers.
- **Customer Responsibilities**
  - Understand **CSP's controls** for hypervisor protection.
  - Check compliance with your **security standards, policies, and regulations**.
- **Reality**
  - CSPs often lack transparency in this area.
  - Customers must place **trust in the CSP** to deliver an isolated, secure guest OS.
- **Key Point** → Security is mostly on the CSP's shoulders, but customers should demand clarity through compliance checks, audits, and contracts.

Threats to the hypervisor

- **1. Integrity and Availability Threats**
- If a **cloud provider's hypervisor** (e.g., VMware ESXi, KVM, or Hyper-V) is compromised, attackers could bring down **all hosted virtual machines (VMs)**.

Example: Imagine a data center hosting hundreds of customer applications. If the hypervisor crashes, all these VMs stop working, causing downtime for banks, hospitals, or e-commerce platforms running on them.

2. Malicious Insider Attack

- *"A vulnerable hypervisor could expose all user domains to malicious insiders."*
  Example: If an administrator at a hosting provider abuses a hypervisor vulnerability, they could gain unauthorized access to customers' VMs.
- For instance, an insider could steal financial data from one tenant's VM or plant malware in another.

3. Subversion Attacks

- Hypervisors can be tricked into running code at a higher privilege level.
  Example: An attacker uploads a malicious VM image to a cloud platform. This VM exploits a flaw in the hypervisor to gain control of the host. Once successful, the attacker can control other tenants' VMs as well.

- **Blue Pill Attack (real research demo)**
- The “Blue Pill” attack (demonstrated by security researcher Joanna Rutkowska) shows how attackers can **install a malicious hypervisor beneath the operating system** without detection.
- Example: Suppose your laptop OS is running normally. **An attacker exploits a CPU virtualization feature to slip in a hidden hypervisor (“**Blue Pill”).
- Your OS then unknowingly runs as a VM inside this hidden hypervisor. The attacker now controls your system *without you realizing it*.

- Downtime threat = all VMs crash.
- Insider threat = admins exploit hypervisor bugs.
- Subversion attack = malicious VM breaks out and takes over.
- Blue Pill = OS gets trapped in a fake hypervisor.

Virtual Server Security

- **Scenario----A company using AWS EC2 (IaaS)**

1. Customer Access & Responsibility

- The company rents EC2 instances (virtual servers) running Linux and Windows.
- AWS provides the **hypervisor isolation** (so one customer's VM can't directly access another's).
- But **the company is responsible** for patching the OS, installing antivirus, configuring firewalls, and securing applications.

  Example: **If the company leaves the VM unpatched, a hacker could exploit vulnerabilities in Windows Server and take control.**

2. Attack Surface – Public Access

- Since virtual servers can be connected to the internet, attackers can attempt brute-force login, exploit weak passwords, or scan for open ports.

Example: An attacker finds port 3389 (RDP for Windows) left open with a weak password → gains admin access.

3. Cloud Service Provider (CSP) Measures

- AWS (or Azure, GCP) often blocks unnecessary ports by default and recommends secure methods like SSH (port 22) or HTTPS for access.
  ☞ Example: Admins should only allow SSH from trusted IP addresses, not the whole internet

Some of the new host security threats in the public IaaS include:

- Stealing keys used to access and manage hosts (e.g., SSH private keys)
- Attacking unpatched, vulnerable services listening on standard ports (e.g., FTP, NetBIOS, SSH)
- Hijacking accounts that are not properly secured (i.e., weak or no passwords for standard accounts)
- Attacking systems that are not properly secured by host firewalls
- Deploying Trojans embedded in the software component in the VM or within the VM image (the OS) itself

Some recommendations:

1. Secure Configuration & Hardened Images

- **What it means:**
- Don’t use a “default” operating system image (like a fresh Windows/Linux install) that comes with **many unnecessary services enabled**.
- Instead, create a **hardened image**: a stripped-down, locked-down version of the OS that has **only what your application needs**.

- **Example Scenario**
- Suppose you're building a **web application** in the cloud:
- **Default way (less secure):**
  - You launch a standard Ubuntu VM from AWS.
  - By default, it may come with packages like FTP, mail services, NFS, and other unused utilities.
  - An attacker can try to exploit vulnerabilities in these unused services, even if you never use them.
- **Hardened way (more secure):**
  - You build a **custom VM image** (base image) that has:
    - Only the **Linux OS kernel**,
    - Your **Apache web server**,
    - And the **database client** needed to connect to your DB.
  - You **remove/disable everything else** (FTP, mail, NFS, printing).
  - Now, even if attackers scan your server, they find only **one entry point (HTTP/HTTPS)** — reducing the chance of attack.

2. Image Integrity & Inventory

- Track and manage **all VM images** and OS versions in use.
- Even if you use an image from the cloud provider (AWS AMI, Azure Marketplace image), subject it to the **same hardening and verification** as your internal servers.
- Best: Maintain your **own enterprise-approved base images**.
- Protect hardened images from unauthorized changes.

3. Key & Credential Management

- Safeguard **private keys** used for SSH or API access.
- **Never embed passwords or API keys** inside VM images.
- Use **encryption/decryption keys** only when needed; keep them isolated from the hosted cloud whenever possible.
- For continuous data-processing applications, minimize risk by restricting key exposure to only runtime use.
- Example: Use AWS KMS, HashiCorp Vault, or Azure Key Vault to manage keys separately instead of hardcoding them in VM images.

4. Authentication & Access Control

- **Disable password-based login** for shell access; use SSH keys instead.
- Require **passwords or MFA** for privilege escalation (sudo, RBAC, SELinux roles).
- Apply **principle of least privilege** for user accounts.

5. Network & Service Hardening

- Run a **host firewall** and open **only the minimum required ports** (e.g., 80/443 for web, 22 for SSH).
- Disable unused network services (FTP, NFS, print services, DB if not needed).

6. Monitoring & Logging

- Enable **system auditing and security event logging**.
- Forward logs to a **dedicated, isolated log server** for tamper-resistance.
- Apply strong access controls to log servers.
- Regularly **review logs for suspicious activity**.

7. Incident Response

- If compromise is suspected:
  - Shut down the affected instance.
  - Perform forensics on a **clean system**, not the compromised one.

Infrastructure Security: The Application Level

1. Why it's critical

- Applications are the **front door** for users and attackers alike.
- Even if the host VM and hypervisor are well secured, a poorly coded or misconfigured application can still be exploited.
- Example: A web app with **SQL Injection** vulnerability allows attackers to steal customer data, even though the VM is hardened.

2. Changing context in the cloud

- Traditional security programs often focused on securing internal, standalone applications.
- Cloud deployment means applications are **exposed to the internet**, so **new risks** must be considered:
  - Multi-tenancy (apps sharing infrastructure with others).
  - Elastic scaling (more components, containers, APIs).
  - Shared responsibility (CSP secures infrastructure, you secure the app).

3. Application security spectrum

- **Standalone Apps** → single user, limited exposure.
- **Enterprise Web Apps** → large-scale, multi-user, internet-facing (e.g., e-commerce, CMS, customer portals).
- Cloud mainly hosts the second type, which has a **larger attack surface**.

4. Development practices

- Many organizations build apps with **web frameworks**: PHP, .NET, J2EE, Ruby on Rails, Python Django/Flask, etc.
- Each framework introduces potential risks if not coded securely:
  - PHP → historically common SQL injection & XSS flaws.
  - .NET/J2EE → misconfigured authentication or insecure deserialization.
  - Python/Ruby → dependency vulnerabilities in 3rd-party packages.

5. Historical shift in attacks

- Before 2007, attackers mainly targeted OS and network vulnerabilities.
- Since then, **web apps became the prime target** because they directly expose sensitive data (financial records, user credentials, business logic).
- Today, most breaches involve application-level exploits like:
  - SQL Injection
  - Cross-Site Scripting (XSS)
  - Insecure Authentication / Session Hijacking
  - API Exploits

Example in practice:

A company hosts an **online shopping app** on AWS.

- If the Linux VM is hardened but the shopping cart application has **poor input validation**, an attacker can inject SQL commands to extract credit card data.
- This shows why **app security must be part of the cloud security program** — not just VM and hypervisor hardening.

XSS(cross-site scripting (XSS)

- Advances in **cross-site scripting (**XSS) and other attacks have demonstrated that criminals looking for financial gain can exploit vulnerabilities resulting from web programming errors as new ways to penetrate important organizations.

- **XSS (Cross-Site Scripting)** is a type of security vulnerability commonly found in web applications.
- It allows attackers to inject malicious scripts into webpages viewed by other users.
- XSS attacks occur when an application includes user input in the output it generates without proper validation or escaping

There are three main types of XSS:

- **Stored XSS (Persistent XSS):**

- The malicious script is permanently stored on the target server (e.g., in a database or a message board).

- When users retrieve the data, the script is executed in their browsers.

Reflected XSS (Non-Persistent XSS):

- The malicious script is reflected off a web server, such as in an error message or search result.
- The attacker needs to trick users into clicking on a maliciously crafted URL to execute the script.

DOM-based XSS:

- The vulnerability lies in the client-side scripts.
- It occurs when the web application's client-side scripts modify the page's Document Object Model (DOM) in an unsafe way.

Browser Security in Cloud Application Security

Why browsers matter

- The **browser is now the main client** for cloud apps (web portals, SaaS dashboards, e-commerce sites, etc.).
- Any compromise in the browser → attacker can steal login credentials, session cookies, or inject malicious code into cloud applications.
- Therefore, **end-to-end security = Cloud App Security + Browser Security**.

Common Browser-Based Threats

1. **Phishing Attacks** – tricking users into entering credentials on fake cloud login pages.
2. **Cross-Site Scripting (XSS)** – attacker injects malicious scripts into a web app, which then execute in the user's browser.
3. **Session Hijacking** – stealing cookies to impersonate a logged-in user.
4. **Malicious Browser Extensions** – capturing keystrokes or stealing tokens.
5. **Drive-by Downloads** – silently installing malware through compromised sites.

Best Practices for Browser Security in Cloud Use

Best Practices for Browser Security in Cloud Use

- **Use HTTPS/TLS everywhere** → prevents eavesdropping and tampering.
- **Enable Content Security Policy (CSP)** → blocks malicious script injection.
- **Regularly patch browsers** → fix known vulnerabilities quickly.
- **Use MFA (multi-factor authentication)** → reduces impact of stolen credentials.
- **Disable weak plugins/extensions** → allow only trusted ones.
- **Enforce secure cookies & SameSite flags** → protect against session hijacking.
- **Educate users** → awareness of phishing & malicious pop-ups.

Application-Level Security Threats

- According to SANS,ll web application vulnerabilities in open source as well as custom-built applications accounted for almost half the total number of vulnerabilities discovered between November 2006 and October 2007.

- The existing threats exploit well-known application vulnerabilities including cross-site scripting (XSS), SQL injection, malicious file execution, and other vulnerabilities resulting from programming errors and design flaws

- Armed with knowledge and tools, hackers are constantly scanning web applications (accessible from the Internet) for application vulnerabilities.

- They are then exploiting the vulnerabilities they discover for various illegal activities including financial fraud, intellectual property theft, converting trusted websites into malicious servers serving client-side exploits, and phishing scams.

- All web frameworks and all types of web applications are at risk of web application security defects, ranging from insufficient validation to application logic errors.

SQL injection

- SQL injection is a type of cyber attack that targets databases through web applications.

- It occurs when an attacker manipulates SQL queries by inserting malicious code into input fields, allowing unauthorized access to the database and potentially leading to data theft, modification, or deletion.

Solution

1. Combination of security controls

- It suggests using **perimeter security controls** (like firewalls, intrusion detection/prevention systems) along with **network- and host-based access controls** (like authentication, authorization, and monitoring at both network and server level).
- **Goal: Protect web applications that are deployed in tightly controlled environments (corporate intranets or private clouds) against external hackers.**

2. Public cloud challenges

- When web applications are deployed in a **public cloud** (instead of private or intranet environments), they face a **higher threat level**.
- They are more exposed to attacks, and hackers may exploit them for **fraudulent or illegal activities**.

3. Security in the cloud (SPI model)

- In the **SPI model** (Software as a Service, Platform as a Service, Infrastructure as a Service), applications must be designed with an **Internet threat model** in mind.
- That means assuming exposure to global threats (DDoS, phishing, malware injection, etc.).
- Security should not be an afterthought—it must be **built into every phase of the Software Development Life Cycle (SDLC)** (planning, design, development, testing, deployment, and maintenance).

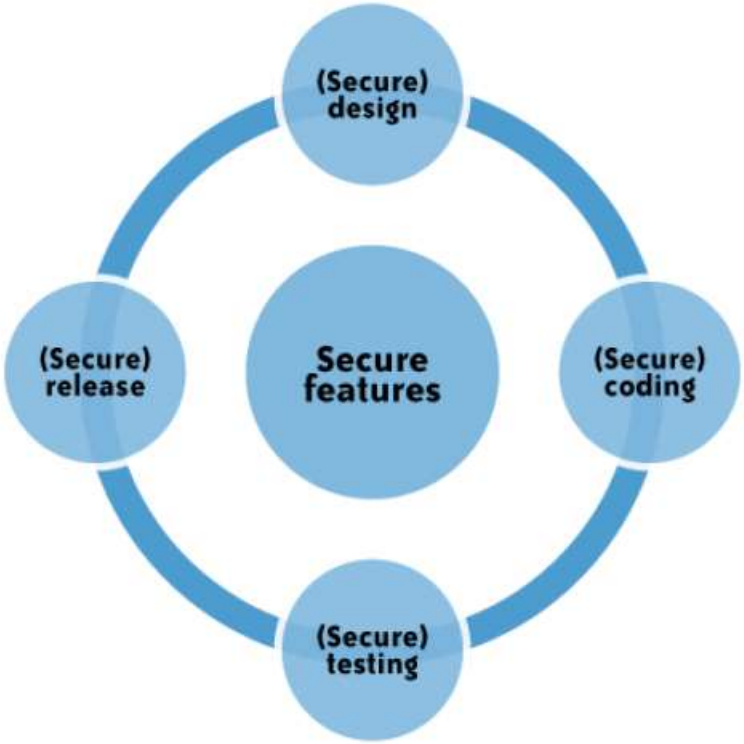


FIGURE 3-2. The SDLC

perimeter security controls

- Perimeter security controls are protective measures designed to safeguard the boundaries of a network or physical location from unauthorized access and external threats.

- These controls act as the first line of defense, ensuring that malicious traffic or attackers are blocked before they can penetrate deeper into the system.

  - **Eg : Firewalls**
  - **Intrusion Detection and Prevention Systems (IDPS):**

DoS and EDoS

1. **Application-level DoS/DDoS attacks**

- These are not just network floods but **application-level attacks**.
- They can **disrupt cloud services for an extended time** by overwhelming servers with fake or excessive requests.

2. **Origin of attacks**

- They usually come from **compromised systems on the Internet**:
  - Hackers hijack vulnerable computers (infected with viruses, worms, or malware).
  - Sometimes, even **unprotected servers** are taken over and used in the attack.
- When multiple hijacked systems are used together, it becomes a **DDoS (Distributed DoS)** attack.

3. **Challenge: malicious traffic vs. real traffic**

- The big problem: these **malicious requests often look very similar to legitimate traffic**.
- This makes it extremely hard to **filter out only the bad traffic** without accidentally blocking real users.

EDOS

Economic angle: EDoS

- Since attackers exploit the **billing structure** of cloud computing, this is not just a denial of service—it's a **denial of sustainability**.
- That's why it's called **EDoS (Economic Denial of Sustainability)**.
- Essentially, the attacker forces the victim to **pay massive cloud bills**, making the service financially unsustainable, even if it doesn't fully crash.

- A normal DoS attack just makes your system unavailable.An EDoS attack makes you go bankrupt by running up your cloud bills (through excessive resource consumption).

End User Responsibility

- Cloud customers are responsible for securing their Internet-connected devices and practicing safe browsing habits.
- **Security Measures**: Essential protection includes using anti-malware, antivirus software, personal firewalls, security patches, and intrusion prevention system (IPS) software.
- **Browser as the Operating System**: Browsers have become key tools for accessing cloud services, making them critical points for security.
- **Vulnerability of Browsers**: Internet browsers are prone to software vulnerabilities that expose users to potential attacks.

- **Browser Hygiene**: To ensure security, users must maintain good browser hygiene, including keeping browsers patched and up to date.
- **Popular Browsers**: Major browsers like Internet Explorer, Firefox, and Safari require frequent updates to protect against security threats.
- **Auto-Update Feature**: Users should enable auto-updates for their browsers to ensure timely installation of security patches.
- **Browser Security Add-ons**: Although browser security add-ons aren't widely available, manual vigilance for updates is essential.

- **Regular Patch Updates**: Customers should regularly check browser vendor websites for security updates and install patches promptly.

- End-to-End Security: Achieving comprehensive security in the cloud involves both maintaining secure browsers and staying informed about potential threats.

Who Is Responsible for Web Application Security in the Cloud?

- Depending on the cloud services delivery model (SPI) and service-level agreement (SLA), the scope of security responsibilities **will fall on the shoulders of both the customer and the cloud provider.**
- The key is to understand what your security responsibilities are versus those of the CSP.
- In that context, recent security surveys have highlighted **the fact that lack of transparency in security controls and practices employed by CSPs is a barrier to cloud adoption.**

- To start with, cloud customers do not have the transparency required in the area of software vulnerabilities in cloud services.
- This prevents customers from managing the operational risk that might come with the vulnerabilities.
- Furthermore, by treating their software as proprietary, CSPs are impeding security researchers from analyzing the software for security flaws and bugs.
- Due to this lack of transparency, customers are left with no choice but to trust their CSPs to disclose any new vulnerability that may affect the confidentiality, integrity, or availability of their application.

SaaS Application Security

- The SaaS model dictates that the provider manages the entire suite of applications delivered to users.
- Therefore, SaaS providers are largely responsible for securing the applications and components they offer to customers.
- Customers are usually responsible for operational security functions, including user and access management as supported by the provider.
- It is a common practice for prospective customers, usually under an NDA, to request information related to the provider’s security practices.
- Some customers go to the extent of hiring independent security vendors to perform penetration testing of SaaS applications (with consent from the provider) to gain assurance independently.
- However, penetration testing can be costly and not all providers agree to this type of verification.

Chapter 2 : Data Security and Storage

- Data-in-transit
- Data-at-rest
- Processing of data, including multitenancy
- Data lineage
- Data provenance
- Data remanence

Aspects of Data Security

1. **Data-in-Transit Security**:
   - The primary risk for data-in-transit is failing to use a well-established encryption algorithm.
   - Using encrypted communication protocols (e.g., FTPS, HTTPS, SCP) ensures both confidentiality and integrity. However, some protocols (e.g., basic FTP or HTTP) can provide confidentiality through encryption but may fail to ensure data integrity.

2. **Data-at-Rest Security**:

- Encrypting data-at-rest is more straightforward in Infrastructure-as-a-Service (IaaS) models (e.g., AWS S3) and is recommended.
- For Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS) models (e.g., Google Apps, Salesforce), encryption can interfere with functionalities like data indexing and searching, making it less feasible.
- Multitenancy in cloud environments (shared infrastructure) means that users' data is commingled, posing potential risks if application vulnerabilities are exploited.

3. **Data Processing Security**:

- Data must be decrypted for processing in the cloud, presenting a security gap where the data is vulnerable during this phase.
- **Fully Homomorphic Encryption (FHE)**, which allows data to be processed while still encrypted, is a breakthrough that could enhance data security in the cloud. However, practical implementation is still in progress due to high computational costs.

Example

- Situation: A medical researcher wants to compute descriptive statistics on a population of lung cancer patients at a hospital.

- Complication: The hospital is unable to share its private medical records with the researcher due to the HIPAA privacy rule.

- Resolution: The hospital encrypts its sensitive data using a fully homomorphic encryption scheme, so that the data is protected while also able to be computed on.
- **Health Insurance Portability and Accountability Act**

- How It Works: The hospital homomorphically encrypts its medical records and sends them to the medical researcher's cloud computing environment.

- Because the data is encrypted, it is fully protected and private in the cloud.

- Next, the researcher runs its analytical functions on the homomorphically-encrypted data in the cloud, manipulating the data while it remains encrypted.

- Last, the researcher downloads the encrypted output, and decrypts the result to reveal the plaintext answer.

- Notice that the sensitive medical record data is encrypted end-to-end, and is only decrypted when revealing the final answer behind organizational firewalls.

4. **Data Lineage and Provenance**:
    - **Data lineage** involves tracking the path of data through the cloud, detailing when and where data is stored or processed. This is important for compliance and audit purposes but difficult to track in public clouds.
    - **Data provenance** goes beyond data integrity, ensuring that the data is computationally accurate and the correct transformations are applied.

5. **Data Remanence**:
    - **Data remanence** refers to residual data left on storage media after attempts to erase it. This poses a risk if sensitive data remains accessible after media is reused or disposed of improperly.
    - Cloud providers often don't give much attention to this risk. The **DoD 5220.22-M** manual and **NIST Special Publication 800-88** provide general guidelines on sanitization and clearing data, though cloud providers may not consistently adhere to them.

Data Security Mitigation

To mitigate data security risks in cloud computing, especially when using public clouds, organizations must adopt several strategies, as outlined below:

1. **Sensitive Data Avoidance in Public Clouds**: The most direct way to mitigate risks is by ensuring that sensitive or regulated data is not stored or processed in a public cloud. If this is unavoidable, organizations should carefully weigh the risks and potentially restrict the cloud's use to **simple storage** purposes only, with strong encryption measures in place.

2. **Encryption for Data-in-Transit and Data-at-Rest**: Organizations should enforce **encryption** for data-in-transit using secure protocols like HTTPS, SCP, or FTPS to safeguard confidentiality and integrity during transmission. For data-at-rest in cloud storage services (e.g., Amazon S3), encrypting it before uploading to the cloud is essential, though challenging with PaaS and SaaS applications due to functionality limitations like indexing or searching.
3. **Enhanced Cloud Service Provider (CSP) Security**: As cryptographic techniques evolve, organizations should stay updated on new developments such as **homomorphic encryption** and **predicate encryption**, which may allow future processing of encrypted data in the cloud without the need for decryption. This could significantly enhance the security of data handled by cloud applications.

4. **Cloud Provider Security Audits**: For customers relying on public clouds, it is important to perform due diligence by checking whether CSPs comply with recognized **security standards**, such as those outlined in **NIST 800-88** for media sanitization and DoD 5220.22-M for data remanence. They should also assess the cloud provider's **data tagging mechanisms** and **application security reviews** to reduce the risk of unauthorized access in a multitenancy environment.
5. **Data Remanence and Deletion Policies**: Ensure the CSP implements stringent policies for the secure deletion of data to avoid **data remanence**, which poses a risk of inadvertent exposure. Organizations should demand compliance with industry best practices and protocols for clearing and sanitizing storage media, especially when dealing with sensitive data.

6. **Custom Cloud Environments for Enhanced Security**: Enterprises handling regulated or highly sensitive data may need to explore **private clouds** or **hybrid cloud environments** where they can have better control over data security measures. Additionally, they may consider paying for premium services, where providers offer stricter security features under a "**regulatory cloud environment.**"
7. **Data Lineage and Provenance**: Although data lineage and provenance are challenging to establish in a cloud environment, these are crucial for regulatory compliance and tracking data integrity. Customers must explore CSP solutions or third-party tools that enhance data tracking and auditability across the data's lifecycle.

Provider Data and Its Security

1. **Metadata Collection and Security**:
   - CSPs gather **metadata** related to your data usage, including how data is accessed, transferred, and stored within their systems. As the volume of your data grows, so does the **value of the metadata**, since it may reveal patterns of data usage, access times, and potentially sensitive operational details.
   - Customers should inquire about the following aspects:
     - What specific metadata does the provider collect regarding your data?
     - How is that metadata **secured**, and what protection measures are in place (e.g., encryption, access controls)?
     - What level of **access** does the customer have to review, monitor, or audit their own metadata?

2. **Provider's Security-Related Data**:
   - **Network-level data**: The CSP is responsible for collecting and protecting data related to firewall activities, **Intrusion Prevention Systems (IPS)**, **Security Information and Event Management (SIEM)** logs, and **router flow data**. These logs are critical for monitoring network activity and detecting potential breaches or anomalies.
   - **Host-level data**: Providers must collect **system logfiles** that capture events happening at the host or server level. This includes any abnormal system behavior that could signal security issues.
   - **Application-level data**: For **SaaS providers**, collecting and protecting **application log data** is vital. This includes logs related to **authentication** and **authorization** processes, which can help detect unauthorized access or privilege escalation attempts.

3. **Incident Response and Digital Forensics**:
    - The security-related data collected by your CSP plays a critical role in **incident response** and **digital forensics**. In the event of a security breach, having access to detailed logs from network, host, and application levels enables both the customer and the provider to quickly identify what happened, contain the issue, and prevent future occurrences.
    - Customers should ensure that their CSP has a clear process for:
        - **Incident reporting**: What mechanisms are in place for reporting security incidents?
        - **Log retention**: How long does the CSP retain these logs for investigation purposes, and is this retention period aligned with your organization's requirements?
        - **Forensic support**: Does the provider offer support for digital forensics in case a breach requires deeper investigation?

4. **Compliance and Audits**:
    - It is essential for customers to understand how their CSP manages and audits security-related data for compliance purposes. **SAS 70**, now superseded by **SOC 1** and **SOC 2** reports, outlines standards for how service providers manage data security, availability, and confidentiality.
    - Providers should regularly undergo independent **security audits** and provide **transparency** on how they collect, store, and protect both customer data and metadata.

Storage

- When storing data in the cloud under the **Infrastructure-as-a-Service (IaaS)** model, such as using storage services like **Amazon S3**, the same fundamental **information security principles** apply: **confidentiality**, **integrity**, and **availability.**
- Each of these must be carefully managed to ensure the protection and security of your data. Here's how these principles apply to cloud-based storage:

1. **Confidentiality**

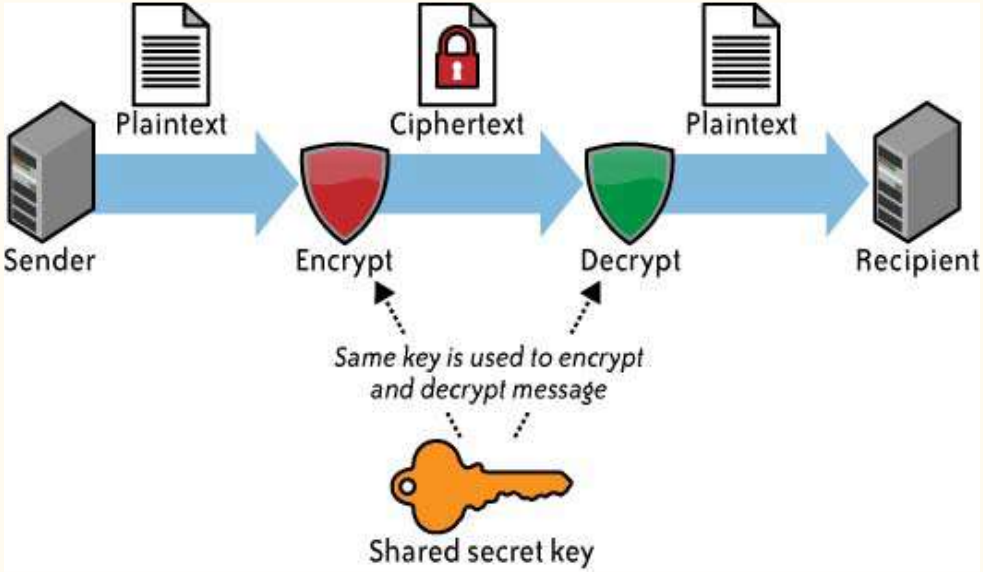
- **Data Encryption**:
  - Cloud providers typically offer encryption mechanisms to safeguard your data. It is crucial that both **data-at-rest** (data stored on disk) and **data-in-transit** (data moving between systems) are encrypted using strong cryptographic methods (e.g., AES-256).
  - Customers should verify whether they have control over **encryption keys** or if the cloud provider manages them. Using **Customer-Managed Keys (CMKs)**, for example, provides an added layer of security and control over data access.
- **Access Controls**:
  - Implement **Identity and Access Management (IAM)** policies to strictly limit who can access sensitive data stored in the cloud. Only authorized users and systems should be granted permission to view, upload, or modify the data.
  - It is also important to use **role-based access controls (RBAC)** to ensure that users only have access to the data they need for their tasks, minimizing the risk of unauthorized access or data breaches.

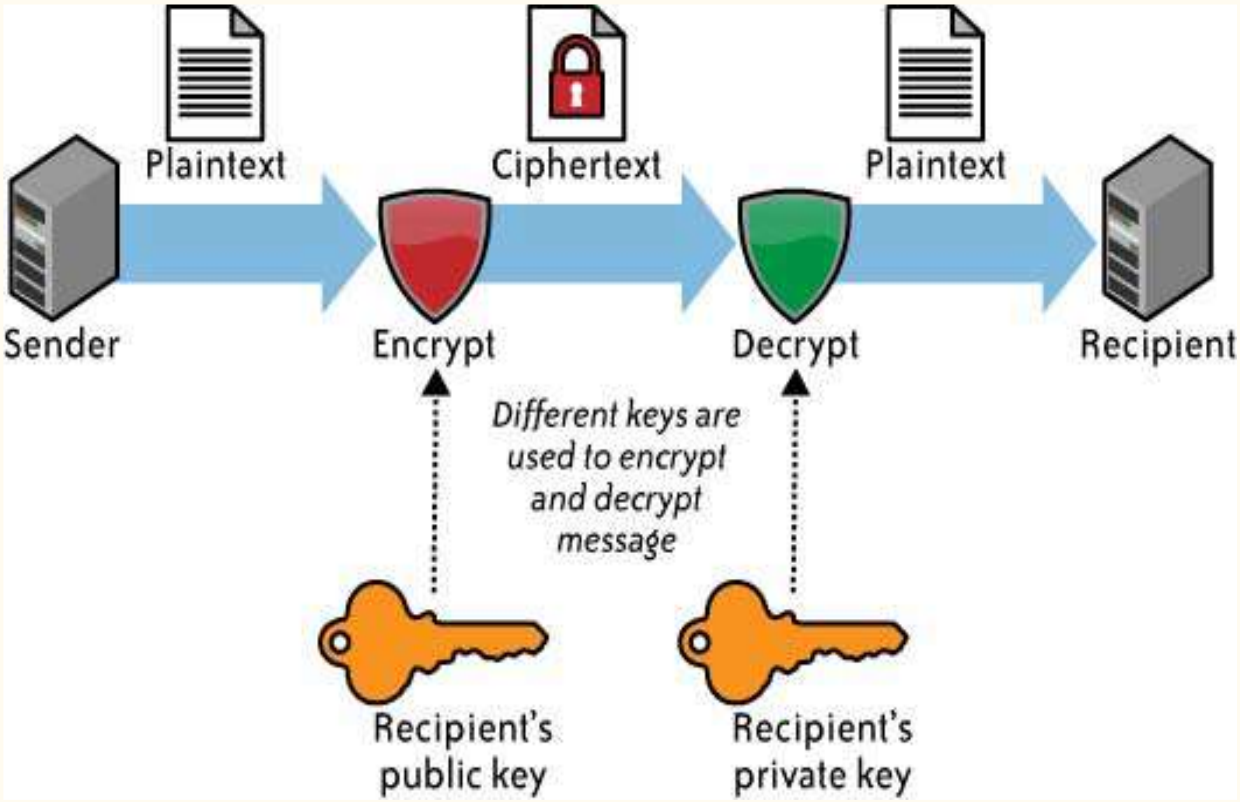
Storage

- Encryption Algorithm
- Key Strength –Length

- It depends, and specifically, it depends on which CSP you are using.
- For example, EMC's MozyEnterprise does encrypt a customer's data.
- However, AWS S3 does *not* encrypt a customer's data.
- Customers are able to encrypt their own data themselves prior to uploading, but S3 does not provide encryption.

- Symmetric encryption involves the use of a single secret key for both the encryption and decryption of data.
- Only symmetric encryption has the speed and computational efficiency to handle encryption of large volumes of data.
- It would be highly unusual to use an asymmetric algorithm for this encryption use case.




Plaintext
Ciphertext
Plaintext
Sender
Encrypt
Decrypt
Recipient
Different keys are used to encrypt and decrypt message
Recipient's public key
Recipient's private key

Key length

- With symmetric encryption, the longer the key length (i.e., the greater number of bits in the key), the stronger the encryption.

- Although long key lengths provide more protection, they are also more computationally intensive, and may strain the capabilities of computer processors.
-
- What can be said is that key lengths should be a minimum of 112 bits for Triple DES (Data Encryption Standard) and 128- bits for AES (Advanced Encryption Standard)—both NIST-approved algorithms.

2. Integrity

- Data Consistency and Accuracy:
  - Data integrity ensures that the data stored in the cloud is accurate and hasn't been altered or tampered with, whether by malicious actors or through system errors.
  - To maintain integrity, cloud storage services often provide features like **checksums** and **hashing algorithms** to detect corruption or unauthorized modifications. For instance, Amazon S3 uses **MD5 checksums** to verify that uploaded objects haven't been altered during transfer.
- Versioning:
  - Enable **versioning** to track changes to files and maintain previous versions of data. This helps restore data to its original state if it becomes corrupted or accidentally modified.
  - Versioning can also support **audit trails**, ensuring that every change to the data can be traced back to a specific user or event, adding a layer of accountability.

3. **Availability**

- **Redundancy and Durability**:
  - Cloud providers ensure high availability and durability by storing data across multiple servers and regions. For example, Amazon S3 guarantees **99.999999999% (11 nines)** of durability, meaning your data is replicated across different locations, significantly reducing the risk of data loss due to hardware failures.
  - Customers should evaluate and choose appropriate **storage classes** based on their availability needs. For example, data that needs constant access should be stored in high-availability storage classes like **S3 Standard**, while archival data can be stored in lower-cost classes like **S3 Glacier**.

Security Best Practices for Cloud Storage:

- **Encrypt sensitive data** both in transit and at rest. Use your own encryption keys for better control if possible.
- Use **multi-factor authentication (MFA)** and strong access controls to prevent unauthorized access to cloud storage resources.
- Enable **logging and monitoring** to keep track of all data access and changes. Cloud services like Amazon S3 can integrate with monitoring tools like **CloudTrail** and **CloudWatch** for this purpose.
- Regularly **audit access logs** and monitor for suspicious activities to quickly respond to potential breaches or unauthorized access attempts.
- Use **data lifecycle management** tools to automate data archiving and deletion processes, ensuring that data is only retained as long as necessary.

MODULE-3

Identity and Access Management

- **THIS CHAPTER PRESENTS THE CURRENT STATE OF THE PRACTICE OF**
- **IDENTITY AND ACCESS** management (IAM) and support for IAM features that aid in Authentication, Authorization, and Auditing (AAA) of users accessing cloud services.

Trust Boundaries and IAM

1. **Traditional Trust Boundaries**:
   - In traditional IT environments, the trust boundary is static, managed by the IT department, and includes networks, systems, and applications.
   - IT uses tools like VPNs, IDS/IPS, and multifactor authentication for security within this boundary.
2. **Impact of Cloud Adoption**:
   - Cloud computing makes trust boundaries dynamic and moves them beyond IT's control, extending into the service provider's domain.
   - This challenges traditional governance models, especially for large enterprises engaged in e-commerce, outsourcing, and supply chains.

3. **Compensation for Network Control Loss**:
    - Organizations need higher-level software controls like application security, user access controls, and identity management to compensate for reduced control over networks.
4. **Key Controls for Cloud Security**:
    - Strong authentication, role-based authorization, identity federation, SSO, user monitoring, and auditing are critical to ensuring security in cloud environments.
5. **Importance of Identity Federation**:
    - Federation enables interaction between systems outside the traditional trust boundary and strengthens trust between organizations and cloud service providers.
    - It supports strong authentication, web SSO, and centralized access control, which are vital for cloud adoption.

6. **Challenges in IAM Practices**:
    - Poorly managed IAM within an organization, such as lack of governance or manual identity management, can weaken cloud security and extend bad practices.
7. **IAM as a Two-Way Street**:
    - Both organizations and cloud service providers must support IAM standards like SAML for federation and compliance with internal policies.
8. **Benefits of Well-Implemented IAM**:
    - Proper IAM practices enhance confidentiality, integrity, and compliance in cloud environments.
    - CSPs supporting IAM standards accelerate cloud adoption and IT migration to cloud services.

SAML (Security Assertion Markup Language)

- **Definition**: SAML is an open standard for exchanging authentication and authorization data between parties, particularly between an identity provider (IdP) and a service provider (SP).
- **Functionality**:
  - **Single Sign-On (SSO)**: Allows users to log in once and gain access to multiple applications without needing to authenticate again.
  - **Assertions**: SAML enables the creation of assertions that convey user identity and attributes from the IdP to the SP.
- **Use Cases**: Commonly used in enterprise environments to enable SSO across web applications and cloud services.

Why IAM?

1. **IAM and Operational Efficiency**:
   - IAM improves efficiency by automating user onboarding and repetitive tasks (e.g., self-service for password resets).
   - Reduces reliance on manual processes such as help desk tickets for common user issues.
2. **IAM for Regulatory Compliance**:
   - Helps protect systems, applications, and data from internal/external threats (e.g., disgruntled employees).
   - Supports compliance with regulatory requirements like HIPAA, SOX, and PCI DSS.
   - Uses frameworks like ISO 27002 and ITIL for implementing access control and operational security.
   - Assists auditors by mapping internal controls to IT controls for managing compliance.

1. **HIPAA (Health Insurance Portability and Accountability Act)**:
    - A U.S. law enacted in 1996 to protect sensitive patient health information.
    - Requires organizations dealing with healthcare data to implement safeguards for the confidentiality, integrity, and availability of protected health information (PHI).
    - Compliance ensures that healthcare organizations secure electronic health records (EHRs), preventing data breaches and unauthorized access.

2. **SOX (Sarbanes-Oxley Act)**:
    - A U.S. federal law passed in 2002 to protect investors by improving the accuracy and reliability of corporate financial reporting.
    - Applies to publicly traded companies and focuses on corporate governance, financial transparency, and internal controls over financial reporting.
    - SOX compliance requires organizations to implement controls for data security and to ensure the integrity of financial systems and processes.

3. **PCI DSS (Payment Card Industry Data Security Standard)**:
   - A set of security standards designed to ensure that all organizations that handle, process, or store credit card information maintain a secure environment.
   - Enforced by major credit card brands (e.g., Visa, MasterCard) and applies to any organization that handles card transactions.
   - Compliance involves securing cardholder data, ensuring strong access controls, and regularly monitoring and testing networks to prevent fraud or breaches.

3. **IAM and New IT Models (Cloud Services)**:
    - IAM enables new IT models like cloud services by facilitating identity federation, linking identity information across trust boundaries.
    - Web SSO and federated user provisioning allow secure interactions between enterprises and CSPs.
4. **Cloud Use Cases Requiring IAM**:
    - Employees and contractors accessing SaaS services using federated identities (e.g., Salesforce).
    - IT administrators provisioning cloud resources using corporate identities (e.g., Amazon EC2).
    - Developers creating accounts for partners on PaaS platforms (e.g., Force.com).
    - End users accessing and sharing cloud storage (e.g., Amazon S3) with access policies.
    - Applications in one cloud accessing resources in another (e.g., Amazon EC2 accessing Mosso storage).

- **Web Single Sign-On (Web SSO)** is a user authentication process that allows users to log in once and gain access to multiple related applications or services without needing to re-enter credentials for each one.

5. **SSO and Rapid Cloud Adoption**:
    - SSO enables businesses to externalize authentication, reducing the time needed to integrate with service providers (e.g., Salesforce.com).
    - IAM enables outsourcing processes securely by extending user access management practices to partners.
6. **Rapid Cloud Service Adoption**:
    - Organizations with strong IAM practices can quickly adopt cloud services while maintaining security and operational efficiency.
    - IAM ensures that security controls remain effective even when services are outsourced or migrated to the cloud.

IAM Challenges

1. **Diverse User Populations**:
   - Organizations manage access for various user groups (employees, contractors, partners), each with different needs and access rights.
   - Changes in roles and responsibilities for business reasons complicate rapid provisioning of access.
2. **Turnover and Staffing Fluctuations**:
   - High user turnover and seasonal changes (e.g., finance or retail) create frequent updates to access rights.
   - Business changes like mergers, acquisitions, outsourcing, and new product releases also affect user access management.

3. **Decentralized Access Policies**:
    - Organizations often apply inconsistent access policies, with multiple directories and disparate identity systems, leading to inefficiencies.
    - This complexity in identity management increases security, compliance, and reputational risks.
4. **Complexity of IAM Initiatives**:
    - Organizations turn to technology solutions to centralize and automate user access management.
    - IAM initiatives are often large, costly, and time-consuming, requiring a balance between business and IT drivers to ensure efficiency and security.
5. **Long-Term Commitment**:
    - Due to the complexity and cost of IAM initiatives, organizations must carefully plan their strategy to address inefficiencies while maintaining effective access control.

IAM (Identity and Access Management) definitions for key functions:

1. **Authentication**:
   - The process of verifying the identity of a user or system.
   - Examples include verifying credentials through protocols like **LDAP** for user ID authentication or service-to-service authentication (e.g., a travel service verifying a credit card through a gateway).
   - Authentication ensures that the entity accessing the system is who they claim to be.
2. **Authorization**:
   - The process of determining what privileges or actions a user or system is allowed to perform after authentication.
   - This step enforces security policies to ensure users or systems only have access to the resources they are permitted to interact with.

- **LDAP (Lightweight Directory Access Protocol)** is an open, vendor-neutral protocol used for accessing and maintaining distributed directory information services over an IP network.
- Directory services provide a centralized way to store information about users, systems, networks, and applications in an organized, hierarchical manner.
- LDAP is widely used for authentication and authorization, particularly in corporate environments.

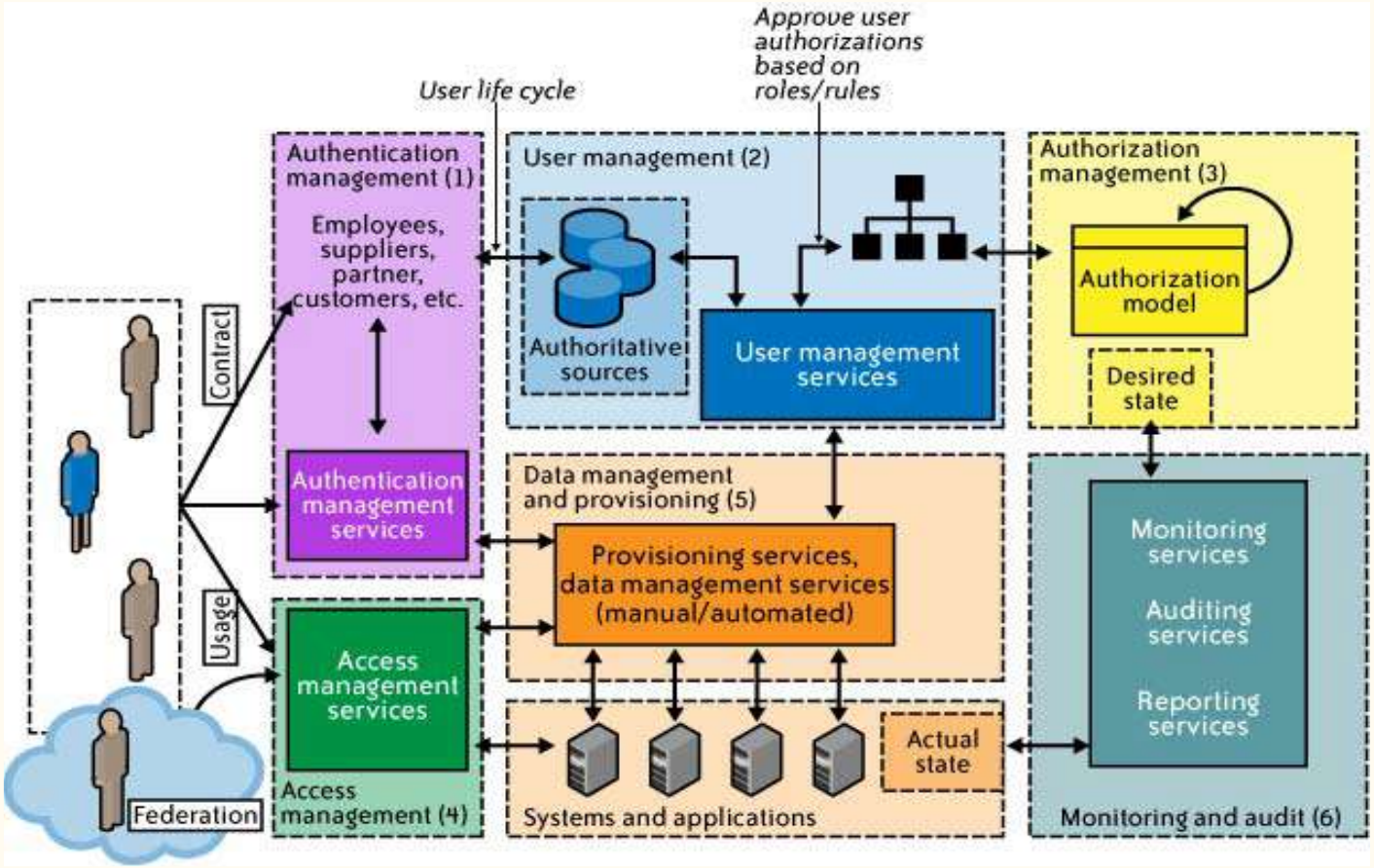
3. **Auditing**:

- Involves reviewing and examining authentication and authorization records.
- It ensures IAM controls are adequate, checks for policy compliance (e.g., separation of duties), detects breaches (e.g., privilege escalation), and recommends countermeasures if needed.

IAM Architecture and Practice

1. **IAM as a Multi-Layered Solution**:
   - IAM is not a simple, monolithic system. It involves a mix of architecture, technology components, processes, and standard practices.
   - Core technology includes a **directory service** (e.g., LDAP, Active Directory) as the central repository for user identities, credentials, and attributes.
   - The directory interacts with other IAM components, such as **authentication**, **user management**, **provisioning**, and **federation services**, to support standard IAM processes.
2. **Multiple Directories**:
   - Organizations may have multiple directories (e.g., Active Directory for Windows systems, LDAP for Unix) due to different environments or through mergers and acquisitions.

3. **Key IAM Processes**:
   - **User Management**: Governs and manages the life cycle of identities within the organization.
   - **Authentication Management**: Governs the process of verifying that an entity is who or what it claims to be.
   - **Authorization Management**: Governs the process of determining what resources an entity can access, based on entitlements and policies.
   - **Access Management**: Enforces access control policies when a user or service requests access to IT resources.
   - **Data Management and Provisioning**: Manages the propagation of identity and authorization data to IT resources, either automatically or manually.
   - **Monitoring and Auditing**: Involves monitoring, auditing, and reporting user compliance with access policies.


Approve user authorizations based on roles/rules
User life cycle
Authentication management (1)
Employees, suppliers, partner, customers, etc.
Authentication management services
User management (2)
Authoritative sources
User management services
Authorization management (3)
Authorization model
Desired state
Contract
Usage
Federation
Access management services
Access management (4)
Data management and provisioning (5)
Provisioning services, data management services (manual/automated)
Systems and applications
Actual state
Monitoring services
Auditing services
Reporting services
Monitoring and audit (6)

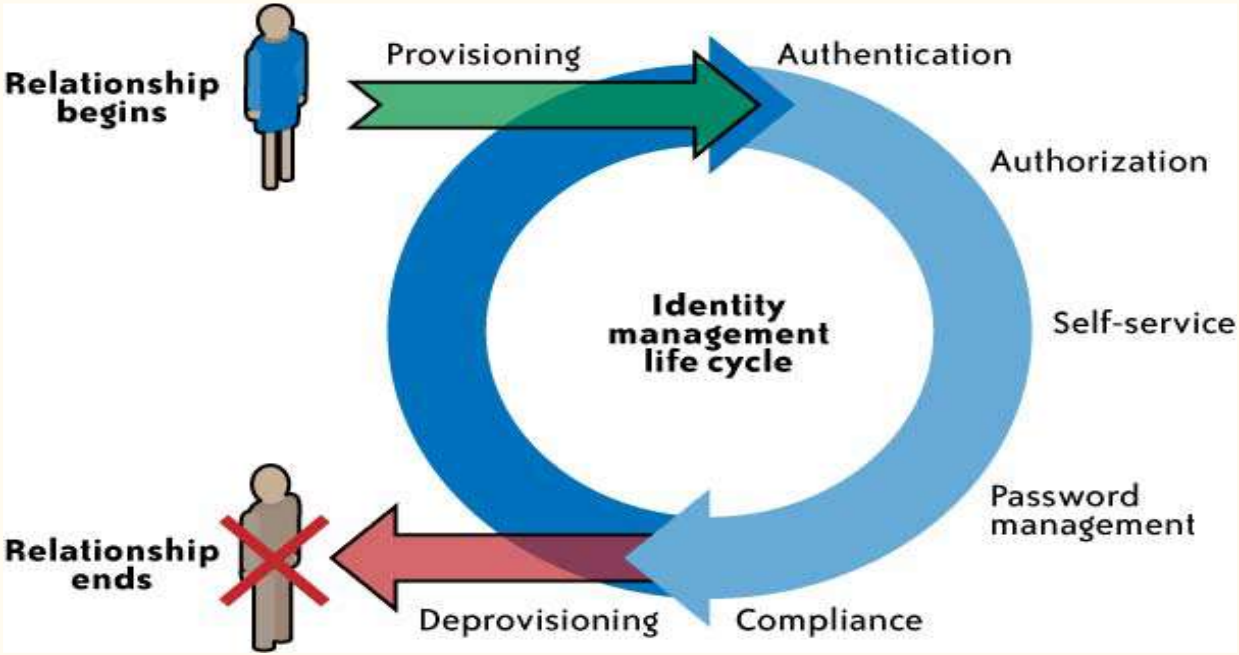
IAM processes support the following operational activities:

1. **Provisioning**:
   - **On-boarding users** to systems and applications by assigning access to data, systems, applications, and databases based on user identity.
   - **Deprovisioning** works in reverse, deactivating or deleting user identities and privileges.
   - It's an essential aspect of resource management, combining IT and HR duties.
2. **Credential and Attribute Management**:
   - **Credential management** involves creating, issuing, managing, and revoking user credentials (e.g., passwords, tokens) to prevent unauthorized access.
   - **Attribute management** manages the life cycle of user attributes and enforces privacy and regulatory handling of these attributes.
   - Ensures credentials are secure both in transit and at rest, including handling expiration and encryption.

3. **Entitlement Management**:
    - **Entitlements** refer to the authorization policies determining user privileges to access systems, applications, and databases.
    - Ensures users have only the necessary privileges for their roles (principle of least privilege), enhancing security for web services, applications, and physical systems.
4. **Compliance Management**:
    - **Monitoring and tracking access rights** to ensure resources are secured and compliant with internal policies and regulations (e.g., segregation of duties, access monitoring).
    - Facilitates periodic auditing and reporting to support regulatory compliance and verifies access through processes like user certification.

5. **Identity Federation Management**:
    - **Federation** involves managing trust relationships between distinct organizations, enabling secure exchange of user and resource information.
    - Supports **Single Sign-On (SSO)** to cloud services, allowing seamless access to external systems without re-authentication.
6. **Centralization of Authentication (authN) and Authorization (authZ)**:
    - A **centralized authentication and authorization infrastructure** removes the need for application developers to build custom security features.
    - Promotes a decoupled architecture, where applications remain agnostic to authentication methods, enhancing flexibility and security through externalized authN and authZ.

Identity life cycle



Getting Ready for the Cloud

1. **User Management Planning**:
   - Organizations need to establish user management processes, including provisioning and deprovisioning user accounts.
   - Enterprises with existing Identity and Access Management (IAM) infrastructure can leverage it for cloud services.
2. **Cloud-Based IAM Solutions**:
   - Companies without an IAM system can use cloud-based solutions from vendors like Symplified, Ping Identity, and TriCipher for identity management.
3. **Federation and SSO**:
   - Identity federation allows secure Single Sign-On (SSO), reducing the need for multiple logins and minimizing security risks.
   - It enables organizations to securely share user identities with trusted cloud service providers (CSPs) without sharing credentials.

4. **IAM Strategy and Architecture**:
    - Organizations should implement an IAM strategy that includes foundational elements like user management and federation capabilities.
    - A consistent user experience and secure access management are achieved through federation.
5. **Identity Provider (IdP)**:
    - Establishing an IdP enables secure identity sharing with CSPs using existing directory services or cloud-based identity management systems.
    - IdPs can integrate with directories like LDAP or Active Directory to manage authentication securely.
6. **Federation Standards**:
    - SAML 2.0 is the de facto standard for identity federation across industries, supporting secure identity sharing in private and public clouds.
    - Other protocols like WS Federation and Liberty Alliance also play roles, but SAML 2.0 is widely adopted.

Relevant IAM Standards and Protocols for Cloud Services

- **IAM Standards and Specifications for Organizations**
  - **Security Assertion Markup Language (SAML)**
  - **Service Provisioning Markup Language (SPML)**
  - **eXensible Access Control Markup Language (XACML)**
  - **Open Authentication (OAuth)**

Four major challenges in user and access management faced by cloud users:

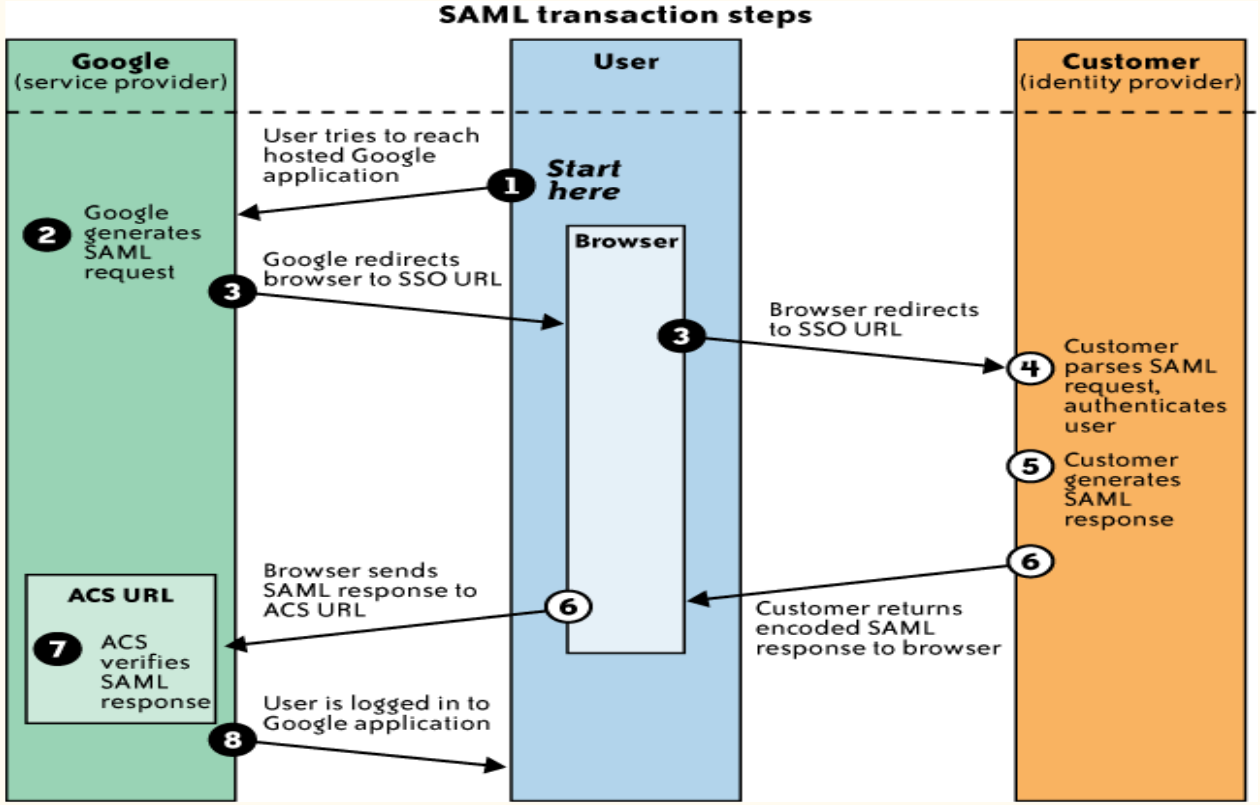
1. How can I avoid duplication of identity, attributes, and credentials and provide a single sign-on user experience for my users? SAML.
2. How can I automatically provision user accounts with cloud services and automate the process of provisoning and deprovisioning? SPML.
3. How can I provision user accounts with appropriate privileges and manage entitlements for my users? XACML.
4. How can I authorize cloud service X to access my data in cloud service Y without disclosing credentials? OAuth.

Security Assertion Markup Language (SAML)

1. **Overview of SAML**:
   - SAML is a mature and widely adopted specification for browser-based federated sign-on.
   - It allows users to access cloud services within a trusted domain after authenticating through an identity service.
2. **Single Sign-On (SSO) Capabilities**:
   - SAML enables Single Sign-On (SSO), allowing users to bypass individual cloud service sign-ons.
   - Once authenticated, users can freely access cloud services without needing separate credentials.
3. **Delegated Authentication**:
   - SAML supports delegation, where the cloud service provider (CSP) delegates authentication to the customer organization's Identity Provider (IdP).
   - This allows the organization to apply its own authentication policies, including risk-based and multifactor authentication.

4. **Strong Authentication and Security**:
    - SAML facilitates strong authentication techniques, such as dual-factor authentication, reducing the risk of phishing attacks.
    - Strong authentication protects user credentials from man-in-the-middle attacks, such as trojans or botnets compromising computers or browsers.

SSO transaction steps using SAML



1. **User Initiates Access**:
    - The user attempts to access a hosted Google application (e.g., Gmail, Start Pages, etc.).
2. **Google Generates SAML Request**:
    - Google creates a SAML authentication request, encodes it, and embeds it into the URL for the organization's Identity Provider (IdP).
    - A **Relay State parameter**, containing the encoded URL of the Google application the user is trying to reach, is also included in the SSO URL.
3. **Google Redirects User**:
    - Google redirects the user's browser to the organization's IdP service with the encoded SAML authentication request.
4. **IdP Decodes SAML Request**:
    - The IdP decodes the SAML request, extracts the URLs for Google's Assertion Consumer Service (ACS) and the user's destination URL (Relay State parameter), and authenticates the user.
    - The IdP can authenticate the user through login credentials or session cookies.

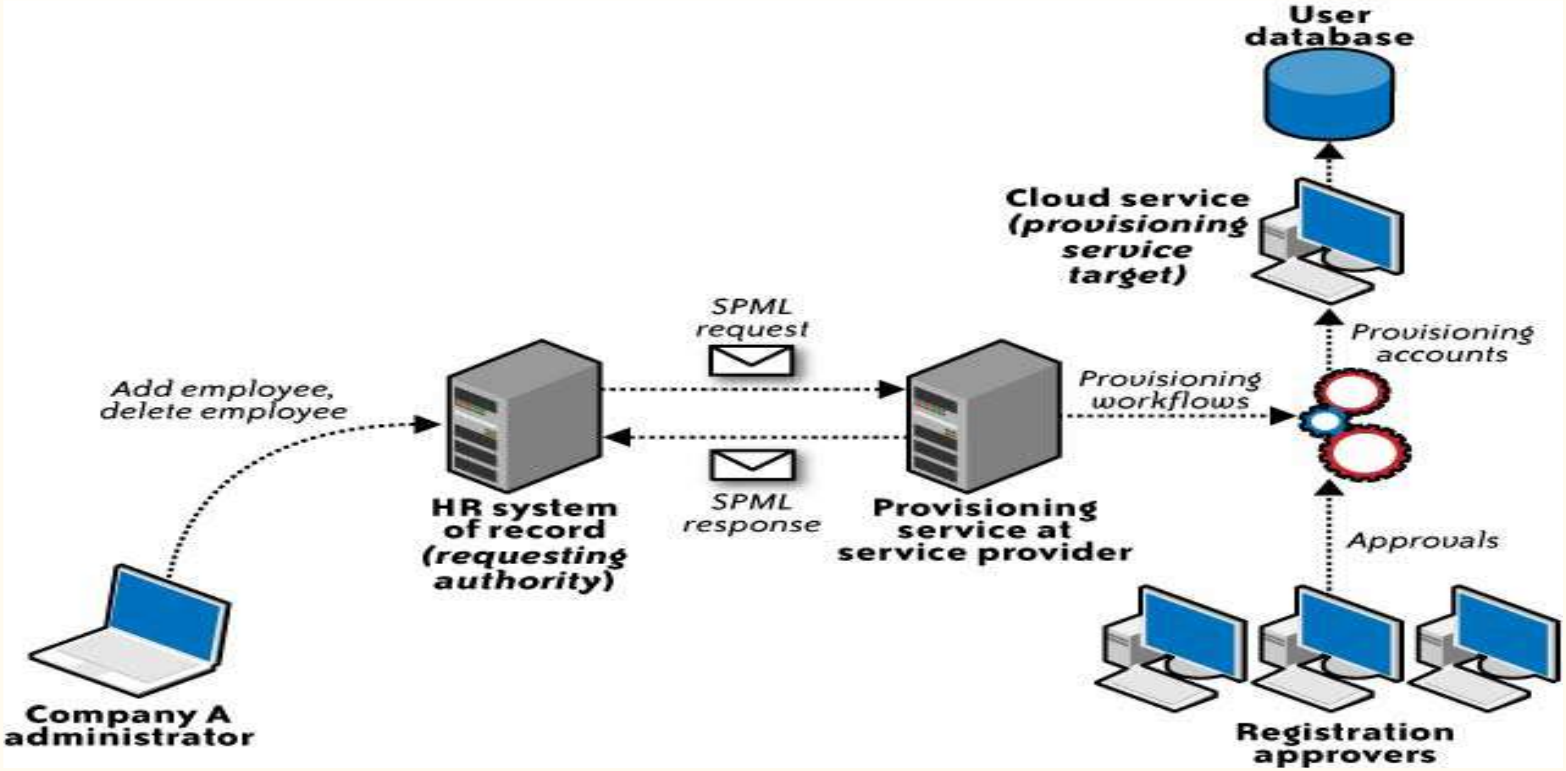
5. **IdP Generates SAML Response**:
    - After successful authentication, the IdP generates a SAML response containing the authenticated user's username, digitally signed using the organization's public/private DSA/RSA keys.
6. **IdP Returns SAML Response**:
    - The IdP encodes the SAML response and the Relay State parameter, then returns them to the user's browser.
    - The browser forwards this information to Google's ACS, either through a form submission (automatically via JavaScript or manually).
7. **Google Verifies SAML Response**:
    - Google's ACS verifies the SAML response using the organization's IdP public key.
    - If verified, Google's ACS redirects the user to the final destination URL.
8. **User is Logged into Google Apps**:
    - The user is now redirected to the destination URL and successfully logged into Google Apps.

SPML Service Provisioning Markup Language

- **SPML Overview**:
  - XML-based framework developed by OASIS.
  - Aims to standardize the exchange of user, resource, and service provisioning information between different systems and organizations.
  - Supports integration and interoperability between various cloud service providers (CSPs).
- **Purpose**:
  - Automates the provisioning, updating, and de-provisioning of user identities for cloud services.
  - Essential for organizations managing large-scale or cross-organizational environments to streamline user access management.
  - Reduces manual intervention in managing user accounts and resource provisioning.

- **Process**:
  - A Cloud Service Provider (CSP) extracts user attributes from a Security Assertion Markup Language (SAML) token.
  - Based on this information, an SPML message is generated in real time.
  - The SPML message is sent to a provisioning service that handles account creation or updates in the cloud service.
  - The provisioning service adds the user identity to the cloud user database, granting access rights automatically.

SPML USE CASE



- **HR System of Record:**
  - Acts as the **requesting authority**.
  - It is an SPML web services client that initiates the provisioning request for a user account (e.g., a new employee or an update to an existing user's account).

- **SPML Provisioning Request:**
  - The HR system generates an SPML request, which contains the necessary user data and provisioning details (e.g., roles, permissions).
  - The request is sent via SPML to the **SPML provisioning service provider** hosted at the cloud service provider.

- **Cloud Service Provider (CSP)**:
  - The cloud service provider acts as the **SPML provisioning service provider**.
  - Upon receiving the SPML request, the CSP processes the request and provisions the necessary user accounts on its cloud services.
- **Provisioning Service Target**:
  - The CSP interacts with its internal provisioning system, the **provisioning service target**, to create or modify user accounts as specified in the SPML request.
  - Once the user account is provisioned, it is recorded in the CSP's **cloud user database**, making the new user eligible for accessing cloud services.

XAML-eXtensible Access Control Markup Language

- **XML-based Access Control Language**:
  - XACML defines an XML schema to express general-purpose access control policies.
  - The policies can be applied to protect any kind of resource, whether it's data, applications, or services.
- **Policy Language and Processing Environment**:
  - XACML provides a **policy language** for writing access control rules, which govern the access rights of users based on roles, attributes, or other criteria.
  - It also includes a **processing model**, which specifies how policies should be managed and how access decisions should be reached.
- **Request/Response Protocol**:
  - XACML specifies a protocol for communication between the **Policy Enforcement Point (PEP)**, which requests access control decisions, and the **Policy Decision Point (PDP)**, which evaluates the policies and responds with an access decision.
  - Both the request for access and the response (allow/deny) are communicated using XML.

eXensible Access Control Markup Language (XACML)

- XACML is an OASIS-ratified, general-purpose, XML-based access control language for policy management and access decisions.

- It provides an XML schema for a general policy language which is used to protect any kind of resource and make access decisions over these resources.

- The XACML standard not only gives the model of the policy language, but also proposes a processing environment model to manage the policies and to conclude the access decisions.

XACML

XACML-eXtensible Access Control Markup Language

- The XACML context also specifies the **request/response protocol** that the application environment can use to communicate with the decision point.
- The response to an access request is also specified using XML.
- In a centrally managed IAM architecture, application-specific authorization models (silos) make it difficult to state the access rights of individual users across all applications.
- Hence,the goal of XACML is to provide a standardized language, a method of access control, and policy enforcement across all applications that implement a common authorization standard.
- These authorization decisions are based on various authorization policies and rules centered on the user role and job function. In short, XACML allows for unified authorization policies.

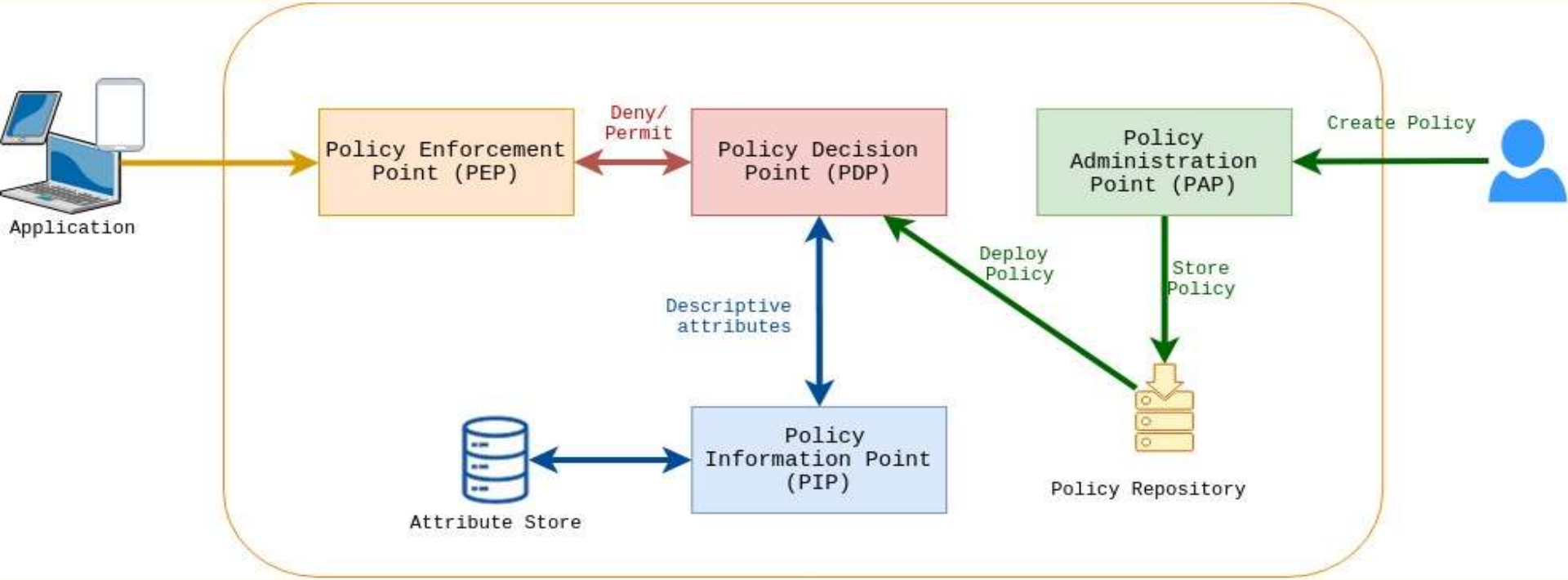
Terminology

- **PAP** – Policy Administration Point Point which manages access authorization policies
- **PDP** – Policy Decision Point Point which evaluates access requests against authorization policies before issuing access decisions
- **PEP** – Policy Enforcement Point Point which intercepts a user's access request to a resource, makes a decision request to the PDP to obtain the access decision
- **PIP** – Policy Information Point, the system entity that acts as a source of attribute values (i.e. a resource, subject, environment)
- **PRP** – Policy Retrieval Point Point where the XACML access authorization policies are stored, typically a database or the filesystem.

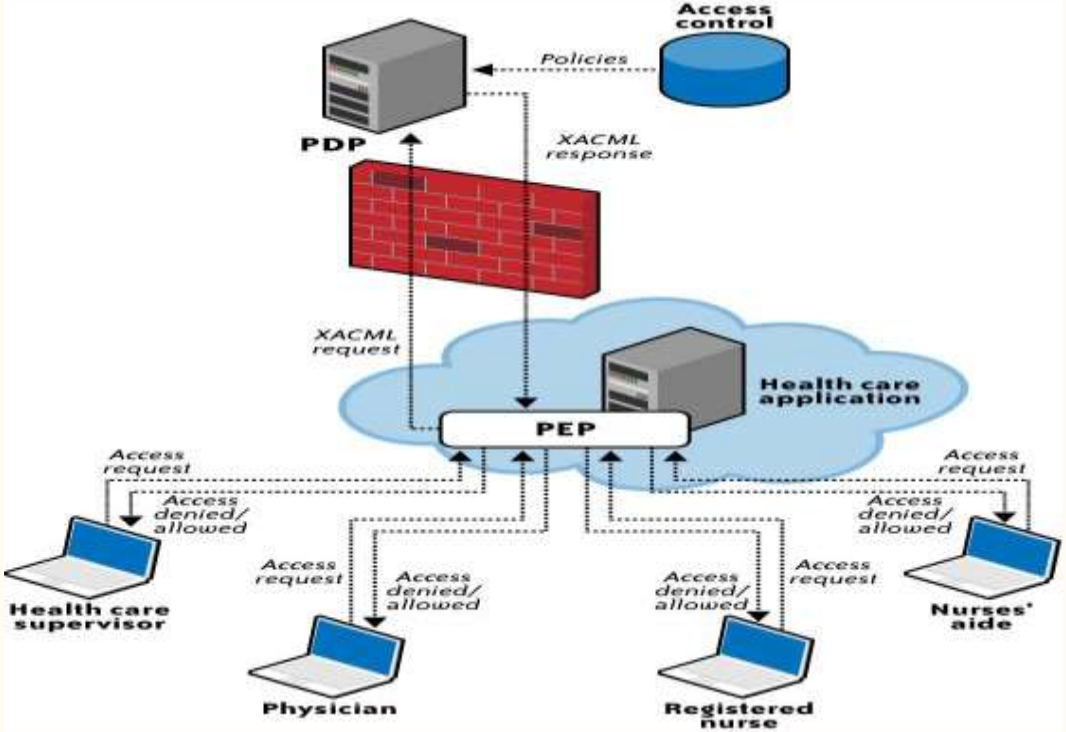
STEPS-

1. The PEP reads a request that was sent by a user and rejects it there.
2. The PEP alters the request such that it is now an XACML request.
3. The request is communicated to the PDP by the PEP.
4. The PDP compares the permission request to the access rules it has defined and decides whether or not the request should be granted.
5. These guidelines are administered by the PAP. Attribute values stored in Policy Information Points may also be accessed if it becomes required.
6. In the end, a judgment is taken at the PDP, and it either returns the value “Permit,” “Deny,” “NotApplicable,” or “Indeterminate” to the PEP in the form of a boolean.

XACML Architecture Diagram



Illustrates the interaction among various health care participants with unique roles (authorization privileges) accessing sensitive patient records stored in a health care application



The figure illustrates the following steps involved in the XACML process:

1. The health care application manages various hospital associates (the physician, registered nurse, nurses' aide, and health care supervisor) accessing various elements of the patient record. This application relies on the policy enforcement point (PEP) and forwards the request to the PEP.
2. The PEP is actually the interface of the application environment. It receives the access requests and evaluates them with the help of the policy decision point (PDP). It then permits or denies access to the resource (the health care record).

3. The PEP then sends the request to the PDP. The PDP is the main decision point for access requests. It collects all the necessary information from available information sources and concludes with a decision on what access to grant. The PDP should be located in a trusted network with strong access control policies, e.g., in a corporate trusted network protected by a corporate firewall.
4. After evaluation, the PDP sends the XACML response to the PEP.
5. The PEP fulfills the obligations by enforcing the PDP's authorization decision.

Open Authentication (OAuth)

- OAuth is an emerging authentication standard that allows consumers to share their private resources (e.g., photos, videos, contact lists, bank accounts) stored on one CSP with another CSP without having to disclose the authentication information (e.g., username and password).

- OAuth is an open protocol and it was created with the goal of enabling authorization via a secure application programming interface (API)—a simple and standard method for desktop, mobile, and web applications.

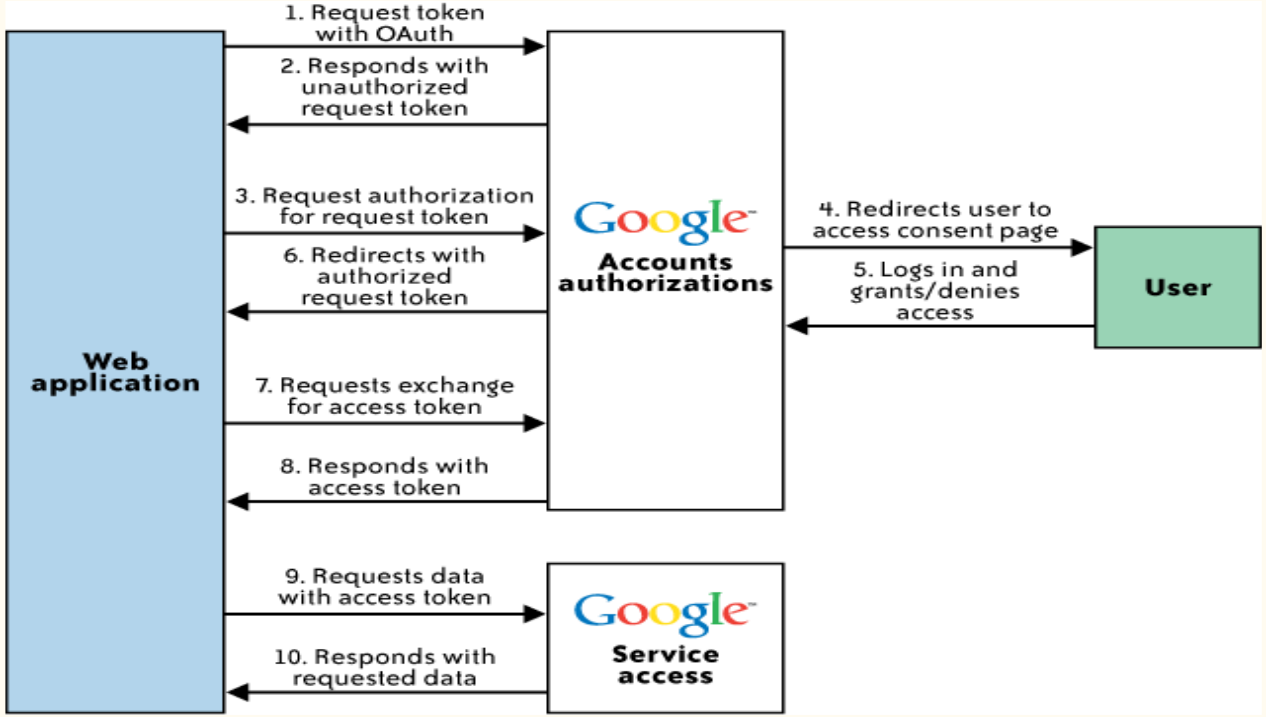
- OAuth is a technological standard that allows you to authorize one app or service to sign in to another without divulging private information, such as passwords.
- If you've ever received a message such as, "Sign in with Facebook?" or "Allow this application to access your account?" you've seen OAuth in action.

- OAuth stands for Open Authorization—not authentication, as it's sometimes assumed to be.
- Authentication is a process that verifies your identity. OAuth does involve your identity, but its purpose is to grant permission to seamlessly connect to you with different apps and services without requiring you to create a new account.
- OAuth provides that simplicity of experience by giving you the option to authorize two apps to share some of your data without revealing your credentials. It strikes a balance between convenience and security.

- For application developers, OAuth is a method for publishing and interacting with protected data.
- For CSPs, OAuth provides a way for users to access their data hosted by another provider while protecting their account credentials.

- Google released a hybrid version of an OpenID and OAuth protocol that combines the authorization and authentication flow in fewer steps to enhance usability.
- Google’s GData API recently announced support for OAuth.

Illustrates the sequence of interactions between customer or partner web application, Google services, and end user:



1. Customer web application contacts the Google Authorization service, asking for a request token for one or more Google service.
2. Google verifies that the web application is registered and responds with an unauthorized request token.
3. The web application directs the end user to a Google authorization page, referencing the request token.
4. On the Google authorization page, the user is prompted to log into his account (for verification) and then either grant or deny limited access to his Google service data by the web application.
5. The user decides whether to grant or deny access to the web application. If the user denies access, he is directed to a Google page and not back to the web application.
6. If the user grants access, the Authorization service redirects him back to a page designated with the web application that was registered with Google. The redirect includes the now-authorized request token.

7. The web application sends a request to the Google Authorization service to exchange the authorized request token for an access token.
8. Google verifies the request and returns a valid access token.
9. The web application sends a request to the Google service in question. The request is signed and includes the access token.
10. If the Google service recognizes the token, it supplies the requested data.

IAM Standards, Protocols, and Specifications for Consumers

- OpenID
- Information cards
- Open Authentication (OATH)
- Open Authentication API (OpenAuth)

OpenID

- OpenID is an open, decentralized standard for user authentication and access control, allowing users to log on to many services with the same digital identity—i.e., a single sign-on user experience with services supporting OpenID.

- As such, it replaces the common logon process that uses a logon username and password, by allowing a user to log on once and gain access to the resources of multiple software systems

- OpenID is primarily targeted for consumer services offered by Internet companies including Google, eBay, Yahoo!, Microsoft, AOL, BBC, PayPal, and so on.

- OpenID adoption for enterprise use (e.g., non-consumer use) is almost nonexistent due to trust issues; some researchers have revealed that OpenID could accelerate phishing attacks that can result in compromising user credentials.

Information cards

- Information cards are another open standard for identity on the Web. The standard itself is directed by the Information Card Foundation, whose steering members include representatives from Google, Microsoft, PayPal, Oracle Novell, and Equifax.

- The Foundations states that its mission is “to reduce the instance of identity theft by securing digital identities in place of traditional logons and passwords.”

- The goal of this standard is to provide users with a safe, consistent, phishing-resistant user interface that doesn’t require a username and password.

- People can use an information card digital identity across multiple sites for convenience without compromising their login information (similar to using an OpenID identity across multiple sites).

- The Information Cards Protocol is designed for use in high-value scenarios, such as banking, where phishing resistance and support for secure authentication mechanisms such as smart cards are critical business requirements.

- Any service provider can implement, issue, or accept information cards (also called i-cards). Information cards are composed using WS-* specifications instead of HTTP redirect, so the specifications are significantly more complicated than OpenID.

- Even though this system offers great protection against identity theft and phishing, it still has a few issues that prevent it from achieving its mission.

- The greatest problem with the system is that it only works if the website used by the consumer is participating and accepts information cards. If this affiliation is not present, the information card is useless.

- As more and more websites accept information cards, the system will become more and more useful, but until that time, they have limited use.

- For example, one can use a managed information card issued by Microsoft Windows Live ID to provide single sign-on to most of Microsoft’s sites, including MSDN, TechNet, Live, and Connect.

Open Authentication (OATH)

- OATH is a collaborative effort of IT industry leaders aimed at providing an architecture reference for universal, strong authentication across all users and all devices over all networks.
- The goal of this initiative is to address the three major authentication methods:

- Subscriber Identity Module (SIM)-based authentication (using a Global System for Mobile Communications/General Packet Radio Service [GSM/GPRS] SIM)
- Public Key Infrastructure (PKI)-based authentication (using an X.509v3 certificate)
- One-Time Password (OTP)-based authentication

Open Authentication API (OpenAuth)

- OpenAuth is an AOL-proprietary API that enables third-party websites and applications to authenticate AOL and AOL Instant Messenger (AIM) users through their websites and applications.

- Using this authentication method, an AIM- or AOL-registered user can log on to a third-party website or application and access AOL services or new services built on top of AOL services.

According to AOL, the OpenAuth API provides the following features:

- A secure method to sign in. User credentials are never exposed to the websites or applications the user signs into.
- A secure method to control which sites are allowed to read private or protected content.
- Automatic granting of permissions only if the user selects Allow Always on the Consent page.
- A prompt for user consent when the website or application attempts to read any private or protected content (e.g., separate consent requests to allow Buddy List information, to send IMs, to read albums).
- Access to other non-AOL websites without the need to create a new user account at each site that supports AOL OpenAuth APIs.

Cloud Identity Administration

- Cloud identity administrative functions should focus on life cycle management of user identities in the cloud—provisioning, deprovisioning, identity federation, SSO, password or credentials management, profile management, and administrative management.

Module-4

Security Management in the Cloud: Introduction, Security Management Standards, Security Management in the Cloud, Availability Management, SaaS Availability Management, PaaS Availability Management, IaaS Availability Management, Access Control, Security Vulnerability, Patch, and Configuration Management.	**8 Hours**

Security Management in the Cloud

- WITH THE ADOPTION OF PUBLIC CLOUD SERVICES, A LARGE PART OF YOUR NETWORK, system, applications, and data will move under third-party provider control.

- The cloud services delivery model will create islands (clouds) of virtual perimeters as well as a security model with responsibilities shared between the customer and the cloud service provider (CSP).

- This shared responsibility model will bring new security management challenges to the organization's IT operations staff.

- With that in mind, the first question a chief information security officer (CISO) must answer is **whether she has adequate transparency from cloud services to manage the governance (shared responsibilities**) and implementation of security management processes (preventive and detective controls) to assure the business that the data in the cloud is appropriately protected

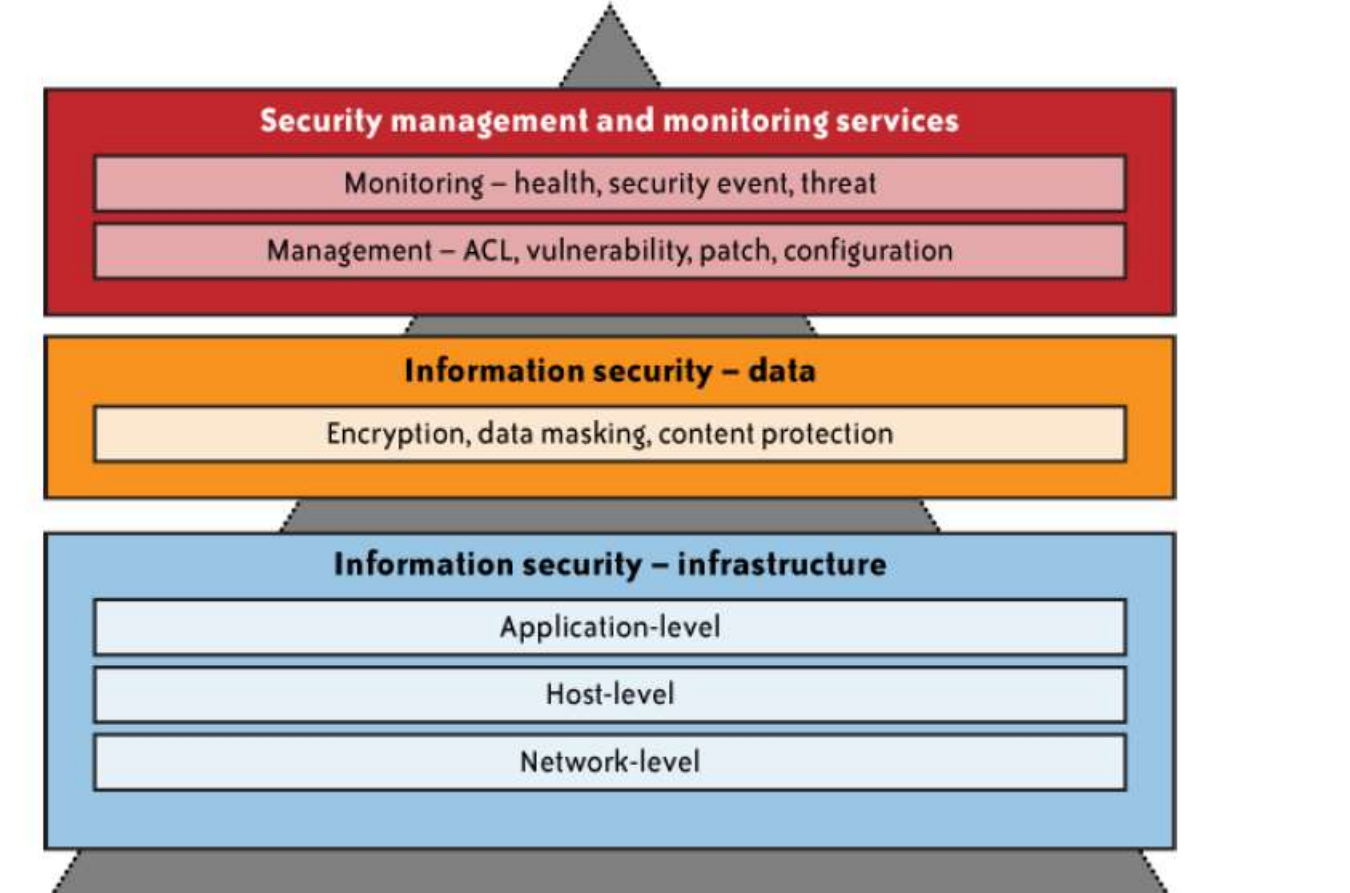
- The answer to this question has two parts: what security controls must the customer provide over and above the controls inherent in the cloud platform, and how must an enterprise's security management tools and processes adapt to manage security in the cloud.

- Both answers must be continually reevaluated based on the sensitivity of the data and the service-level changes over time.

USER -Responsibility

- As a customer of the cloud, you should start with the exercise of understanding the **trust boundary of your services in the cloud**.
- You should understand all the layers you own, touch, or interface with in the cloud service—network, host, application, database, storage, and web services including identity services .
- You also need to understand the scope of IT system management and monitoring responsibilities that fall on your shoulders, including access, change, configuration, patch, and vulnerability management.

Security management and monitoring scope



Security management and monitoring scope

- Mature IT organizations are known to employ security management frameworks, such as ISO/ IEC 27000 and the Information Technology Infrastructure Library (ITIL) service management framework.
- These industry standard management frameworks provide guidance for planning and implementing a governance program with sustaining management processes that protect information assets

- For example, **ITIL gives a detailed description of a number of important IT practices with comprehensive checklists, tasks, and procedures that can be tailored to any IT organization.**
- A key tenet of ITIL, and one that is applicable to cloud computing, is that organizations (people, processes) and information systems are constantly changing.
- Hence, management frameworks such as ITIL will help with the continuous service improvement that is necessary to align and realign IT services to changing business needs.
- Continuous service improvement means identifying and implementing improvements to the IT services that support business processes such as sales force automation using a cloud service provider.

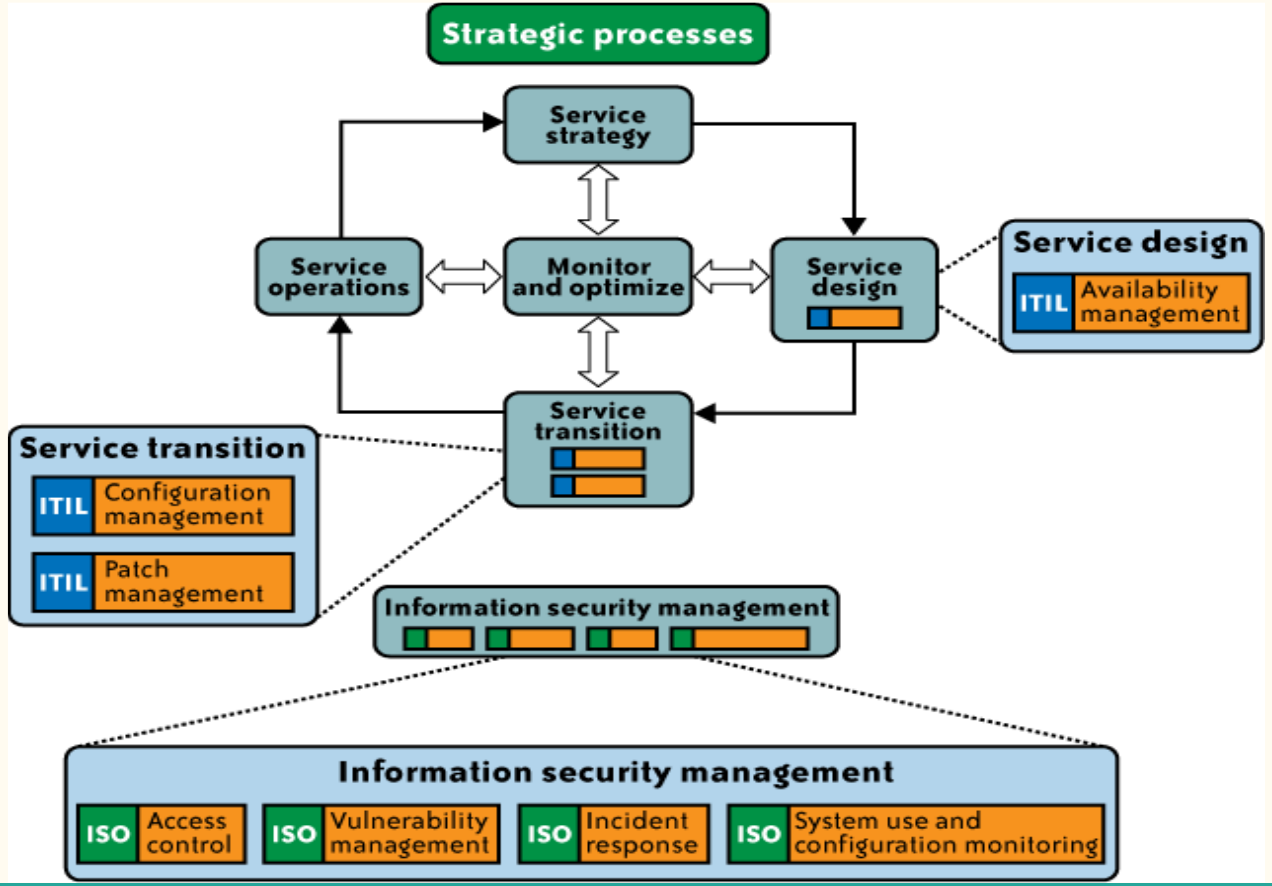
Realization of security requirements

Security requirements are usually defined in the SLA as well as in other external requirements, which are specified in underpinning contracts, legislation, and internally or externally imposed policies.

Realization of a basic level of security

This is necessary to guarantee the security and continuity of the organization and to reach simplified service-level management for information security management.

The ITIL life cycle in a enterprise



Security Management Standards

- ITIL
- ISO 27001/27002

- The Information Technology Infrastructure Library (ITIL) is a set of best practices and guidelines that define an integrated, process-based approach for managing information technology services.
- ITIL can be applied across almost every type of IT environment including cloud operating environment.
- ITIL seeks to ensure that effective information security measures are taken at strategic, tactical, and operational levels.
- Information security is considered an iterative process that must be controlled, planned, implemented, evaluated, and maintained.

ITIL breaks information security down into:

Policies

The overall objectives an organization is attempting to achieve

Processes

What has to happen to achieve the objectives

Procedures

Who does what and when to achieve the objectives

Work instructions

Instructions for taking specific actions

ISO 27001/27002

- ISO/IEC 27001 formally defines the mandatory requirements for an Information Security Management System (ISMS).
- It is also a certification standard and uses ISO/IEC 27002 to indicate suitable information security controls within the ISMS.
- However, since ISO/IEC 27002 is merely a code of practice/guideline rather than a certification standard

The ITIL, ISO/IEC 20000, and ISO/IEC 27001/27002 frameworks help IT organizations internalize and respond to basic questions such as:

- How do I ensure that the current security levels are appropriate for your needs?
- How do I apply a security baseline throughout your operation?
- To that end, they help you to respond to the question: how do I ensure that my services are secure?

Security Management in the Cloud

- After analyzing the management process disciplines across the ITIL and ISO frameworks, we (the authors) identified the following relevant processes as the recommended security management focus areas for securing services in the cloud

  - Availability management (ITIL)
  - Access control (ISO/IEC 27002, ITIL)
  - Vulnerability management (ISO/IEC 27002)
  - Patch management (ITIL)
  - Configuration management (ITIL)
  - Incident response (ISO/IEC 27002)
  - System use and access monitoring (ISO/IEC 27002)