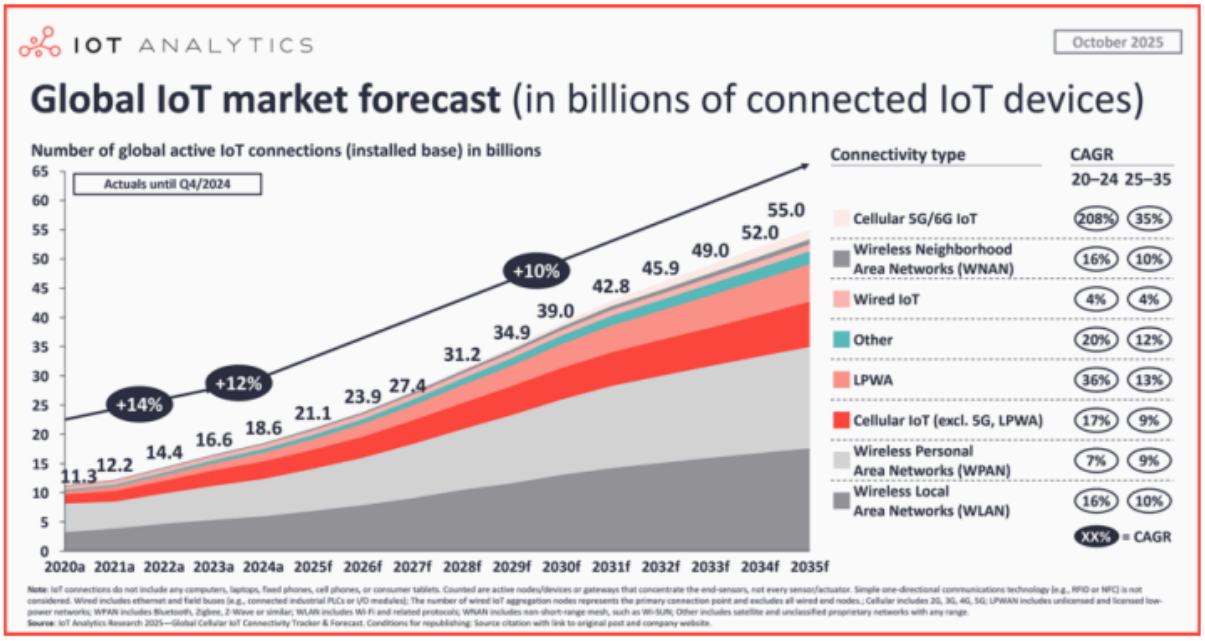
IoT_and_Cloud

#week8 #security

Internet of Things

Coined in 1999, describes any device that interacts with the world around it by gathering data from sensors or providing real world interactions via actuators. Generally these devices are connected to each other or to the internet

IoT Sensors



IoT Devices

Production/Commercial Devices

Custom-made: Designed for specific tasks and environments.

Examples: Fitness trackers, industrial machine controllers.

Characteristics:

- Custom circuit boards for size and functionality.
- Ruggedness for harsh environments.

Developer Kits

General-purpose IoT devices for learning and prototyping.

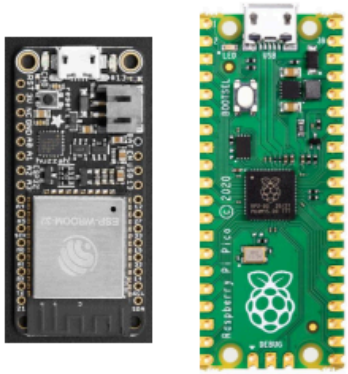
Characteristics

External pins for sensor/actuator connection.

Debugging hardware.

Additional resources (often costly for mass production).

Types of Developer Kits

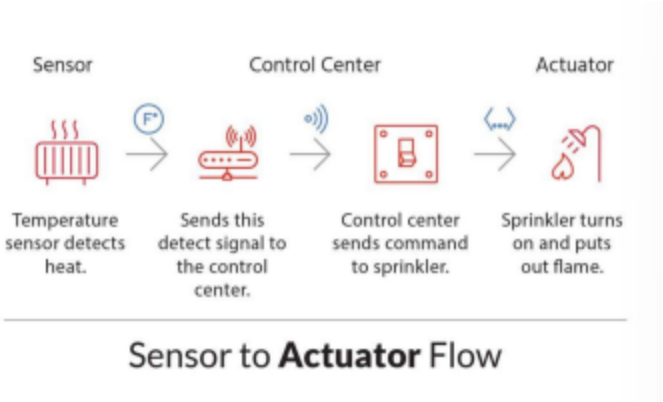


Microcontrollers: Smaller, focused on processing and control.

Single-board computers: More powerful, include RAM, storage, etc.

Microcontrollers are small computers consisting of one or more cpus, memory, and programmable i/o connections

Single-board computers are small computing devices that contain all the elements of a complete computer, these have closer specifications to a desktop or laptop pc but are substantially cheaper

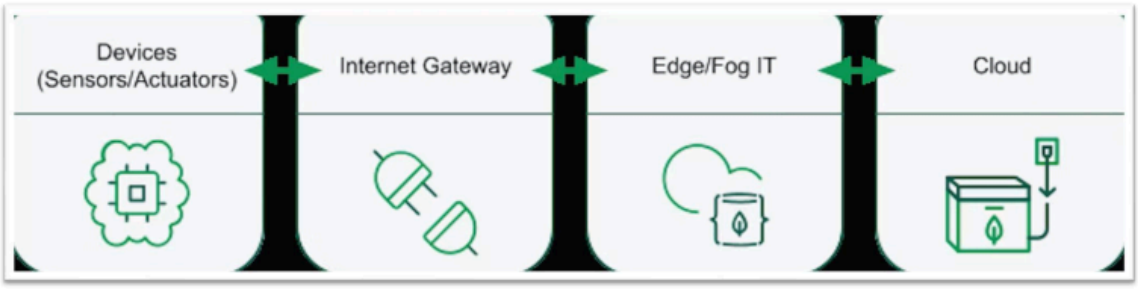


One core aspect of IoT is machine to machine(M2M) communication, this is where billions of smart and autonomous devices connect to the internet to leverage AI, cloud technologies, and big data analytics.

IoT Architecture

IoT typically consists of four layers:

- Perception Layer
- Network Layer
- Processing Layer
- Application Layer



Can also be displayed this way

- Devices: Data produced by sensors is transmitted to internet gateway stage
- Internet gateways: Receives raw data from IoT devices and pre-process before sending to cloud
- Edge: Once data is pre-process the aim is to fully process the data asap

Security Challenges

1. Device-Level Vulnerabilities

- ***Constrained Resources:*** Many IoT devices have limited computational power, memory, and battery life, making it difficult to implement robust security measures like encryption or regular security updates.
- ***Insecure Default Configurations:*** Devices often ship with default passwords, unnecessary services enabled, and insecure settings. The Mirai botnet attack exploited exactly this vulnerability, compromising hundreds of thousands of devices.
- ***Lack of Update Mechanisms:*** Many IoT devices lack secure, automated update capabilities. Some devices are never updated throughout their operational lifetime, leaving known vulnerabilities unpatched.
- ***Physical Security:*** IoT devices are often deployed in unsecured locations, making them vulnerable to physical tampering, device theft, or hardware attacks.

2. Network-Level Threats

- ***Man-in-the-Middle Attacks:*** Unencrypted communication channels allow attackers to intercept, modify, or inject malicious data.
- ***Protocol Vulnerabilities:*** Many IoT protocols (Zigbee, Z-Wave, Bluetooth LE) have inherent security weaknesses or implementation flaws.
- ***Network Segmentation Issues:*** IoT devices on the same network as critical systems create lateral movement opportunities for attackers.

3. Data Privacy and Integrity

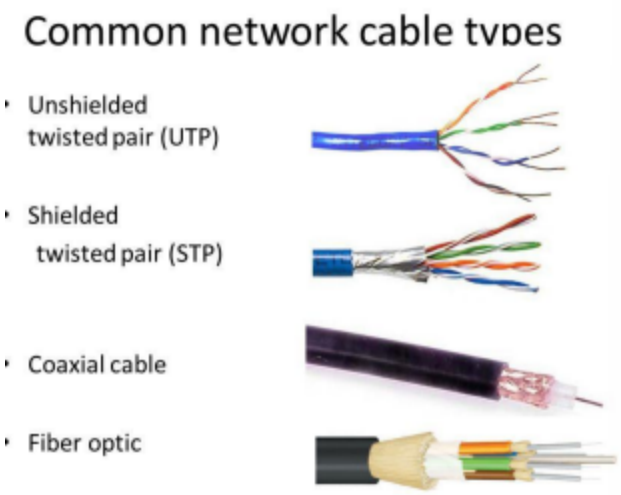
- ***Sensitive Data Exposure:*** IoT devices collect vast amounts of personal data (location, health metrics, behaviour patterns) that may be inadequately protected.
- ***Data Tampering:*** Without proper integrity checks, sensor data can be manipulated, leading to incorrect decisions in critical systems.

4. Scalability and Management

- With billions of IoT devices deployed globally, managing security at scale becomes extremely challenging. Traditional security tools aren't designed for the heterogeneity and volume of IoT environments.

Wired vs Wireless Connectivity

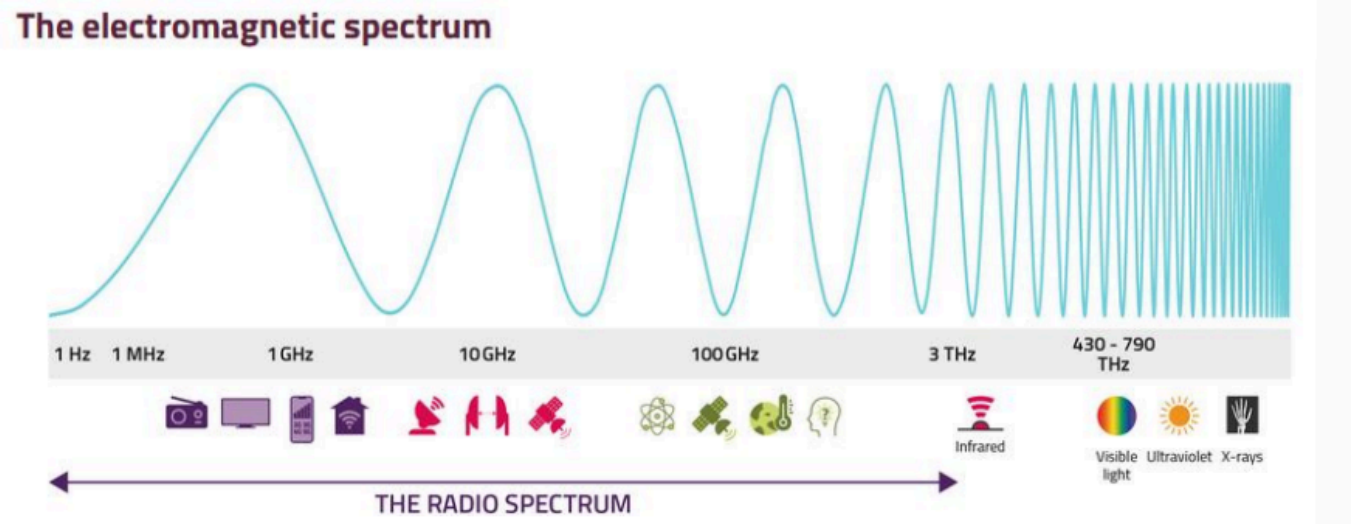
Wired connectivity is when electrical signals are sent over physical wires between devices, therefore signals are contained withing the physical table.



Wireless Connectivity is typically radio or optical and are broadcasted across the environment between IoT devices, here the carrier waveform is launched as electromagnetic waves intro free space from one device to another. This could be sent over an antenna that radiates energy in a specific direction, which is then recieved by another antenna

Wireless IoT

- **Wi-Fi**: This is a wireless networking technology that is widely used in homes and businesses. IoT devices can connect to Wi-Fi networks to transmit data to the internet.
- **Cellular**: Cellular connectivity allows IoT devices to connect to the internet using cellular networks, such as 4G and 5G. This technology is particularly useful for IoT devices that are deployed in remote locations where Wi-Fi is not available.
- **Bluetooth**: Bluetooth is a short-range wireless technology that is commonly used for connecting devices such as smartphones, headphones, and speakers. IoT devices can also use Bluetooth to communicate with each other.
- **Zigbee**: Zigbee is a wireless protocol that is designed specifically for low-power IoT devices. It is often used in home automation systems and other applications where battery life is important.
- **LoRaWAN**: LoRaWAN is a long-range, low-power wireless protocol that is used for IoT devices that need to transmit data over long distances.
- **Ethernet**: Ethernet is a wired networking technology that is commonly used in industrial settings. IoT devices can connect to Ethernet networks to transmit data to the cloud.



IoT Best Practices

1. Secure Device Design

- **Security by Design:** Integrate security considerations from the initial design phase
- **Hardware Security:** Implement **Trusted Platform Modules** (TPM), secure boot, and hardware encryption
- **Minimal Attack Surface:** Disable unnecessary services, ports, and features
- **Secure Default Configurations:** Force password changes upon first use, disable default accounts

2. Authentication and Access Control

- **Strong Authentication:** Multi-factor authentication where feasible
- **Device Identity:** Unique cryptographic identities for each device
- **Least Privilege Principle:** Grant minimum necessary permissions
- **Certificate-Based Authentication:** Use PKI infrastructure for device authentication

3. Secure Communication

- **End-to-End Encryption:** TLS/DTLS for data in transit
- **Lightweight Cryptography:** Algorithms optimized for resource-constrained devices (e.g., ECC, ChaCha20)
- **Secure Protocols:** Use secure versions of communication protocols

4. Security Monitoring and Updates

- **Continuous Monitoring:** Real-time anomaly detection and behavioural analysis
- **Secure Update Mechanisms:** Signed firmware updates, rollback capabilities
- **Patch Management:** Regular security patches throughout device lifecycle
- **Logging and Auditing:** Comprehensive logs for forensic analysis

5. Network Security

- **Network Segmentation:** Isolate IoT devices on separate VLANs
- **Firewall Rules:** Strict ingress/egress filtering
- **Intrusion Detection:** Deploy IDS/IPS systems tuned for IoT traffic patterns

The OWASP IoT top 10 reasons for weak IoT security is as follows:

Weak, guessable, or hardcoded passwords
Insecure network services
Insecure ecosystem interfaces
Lack of secure update mechanism
Use of insecure or outdated components
Insufficient privacy protection
Insecure data transfer and storage
Lack of device management
Insecure default settings
Lack of physical hardening

Cloud Security

Cloud services providers include the following

1. Infrastructure as a Service (IaaS)

Provider manages: Physical infrastructure, virtualization

Customer manages: Operating systems, applications, data

Examples: AWS EC2, Azure Virtual Machines, Google Compute Engine

Security responsibility: Primarily customer-managed

2. Platform as a Service (PaaS)

Provider manages: Infrastructure, runtime environment, middleware

Customer manages: Applications, data

Examples: AWS Elastic Beanstalk, Azure App Service, Google App Engine

Security responsibility: Shared between provider and customer

3. Software as a Service (SaaS)

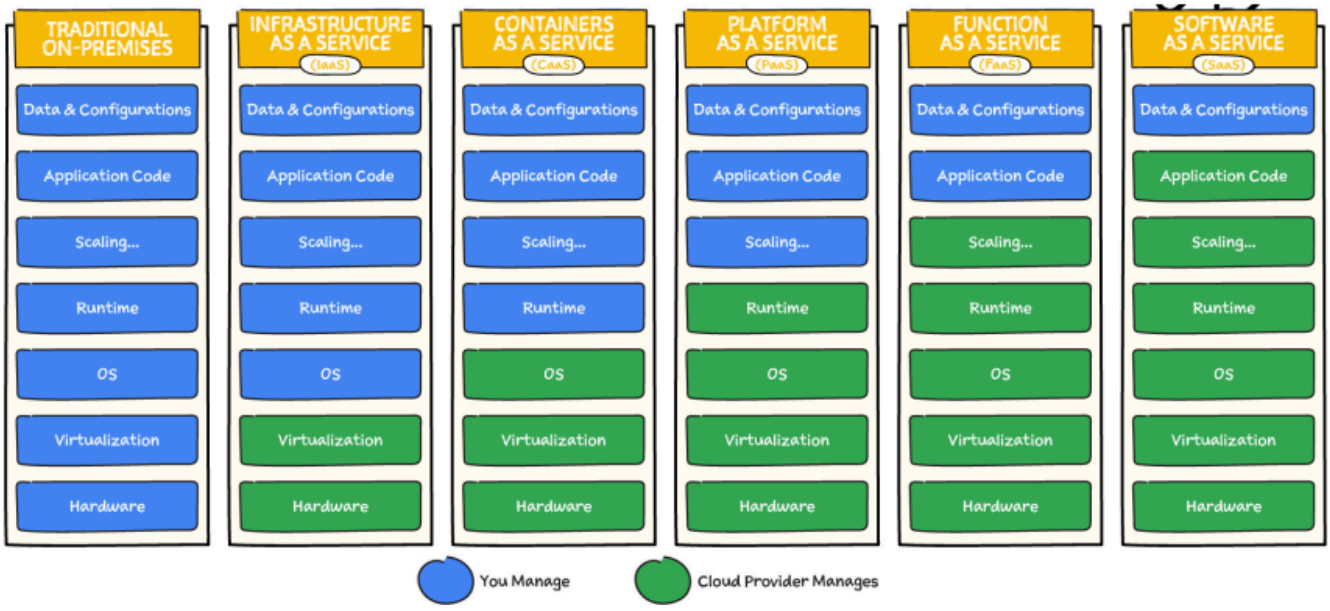
Provider manages: Everything except user data and access

Customer manages: User access, data configuration

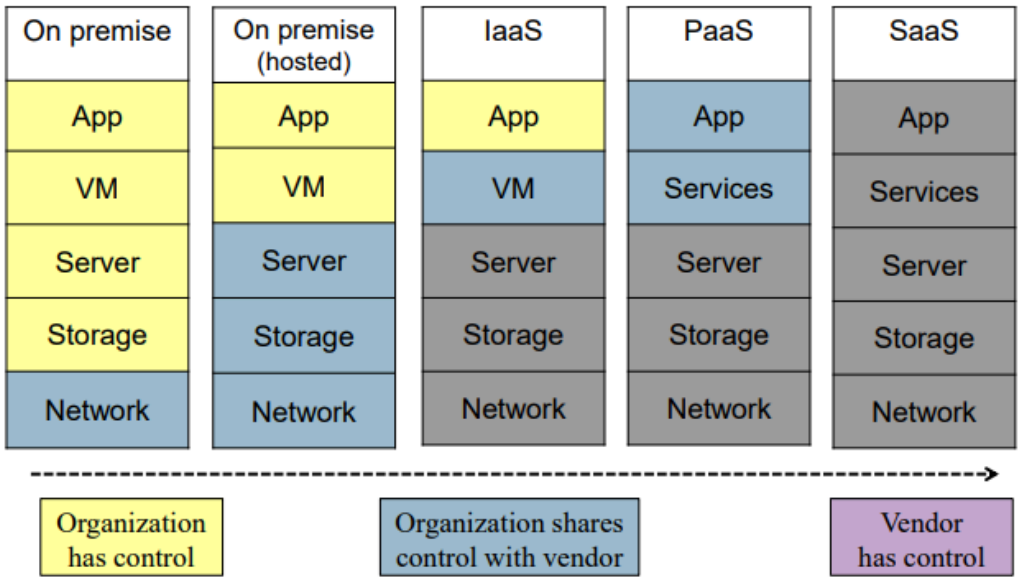
Examples: Salesforce, Microsoft 365, Google Workspace

Security responsibility: Primarily provider-managed

IaaS, PaaS, SaaS, and CaaS



Control, liability and accountability



Deployment Models

- **Public Cloud:** Multi-tenant, cost-effective, managed by third party
- **Private Cloud:** Single-tenant, greater control, higher cost
- **Hybrid Cloud:** Combination of public and private
- **Multi-Cloud:** Using multiple cloud providers

Cloud Security Challenges

1. **Shared Responsibility Model**
   - A fundamental concept in cloud security is understanding the division of security responsibilities between the cloud service provider (CSP) and the customer. Misunderstanding this model is a primary cause of cloud security incidents.
   - CSP is responsible for security *of* the cloud (infrastructure, facilities, hardware)
   - Customer is responsible for security *in* the cloud (data, applications, access management)
2. **Data Security and Privacy**
   - *Data Breaches:* Cloud environments store massive amounts of data, making them attractive targets. Misconfigurations are the leading cause of cloud data breaches.
   - *Data Sovereignty:* Data may be stored in multiple geographic locations, potentially violating regulatory requirements (GDPR, HIPAA, data localization laws).
   - *Data Residency and Compliance:* Organizations must ensure data storage and processing comply with industry regulations and geographic requirements.
   - *Multi-Tenancy Risks:* In public clouds, multiple customers share the same physical infrastructure, raising concerns about data isolation and side-channel attacks.
3. **Identity and Access Management (IAM)**
   - *Compromised Credentials:* Stolen or weak credentials are a primary attack vector in cloud environments.
   - *Privilege Escalation:* Attackers exploiting misconfigurations to gain higher-level permissions.
   - *Inadequate Access Controls:* Overly permissive policies violating the principle of least privilege.

Cloud Security Challenges

4. **Misconfigurations**
   - According to recent studies, misconfiguration is responsible for over 60% of cloud security incidents:
   - Publicly exposed storage buckets (AWS S3, Azure Blob)
   - Overly permissive security groups and firewall rules
   - Disabled security features and logging
   - Unencrypted data storage
5. **Insecure APIs**
   - Cloud services are accessed via APIs, which can be vulnerable to:
   - Authentication and authorization flaws
   - Injection attacks
   - DDoS attacks
   - Data exposure through verbose error messages
6. **Account Hijacking**
   - Attackers gaining unauthorized access through:
   - Phishing attacks
   - Credential stuffing
   - Exploiting API vulnerabilities
   - Social engineering

7. Insider Threats

- Malicious or negligent insiders with legitimate access can:
- Exfiltrate sensitive data
- Misconfigure security settings
- Delete critical resources
- Abuse elevated privileges

8. Vendor Lock-In and Supply Chain

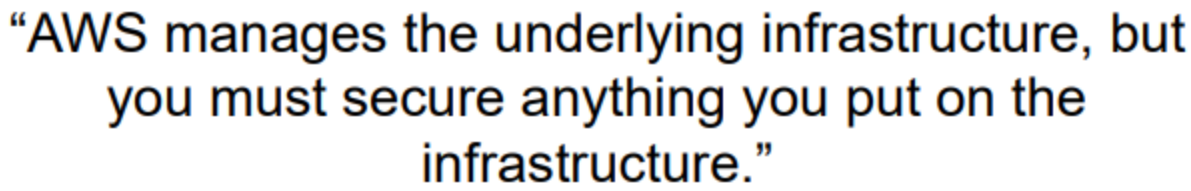
- Dependency on specific cloud provider technologies
- Difficulty migrating between providers
- Third-party risks through cloud supply chain

Security management in AWS consists of the following:

- **Availability**, low availability leads to outages in sits. To protect against this AWS features DDoS protection and fault-tolerant independent failure zones
- **Access Control**, who should have access? AWS has built in firewalls and MFA
- **Monitoring**, with AWS you can monitor availability and unauthorised activities with tools such as DoS, MITM, port scanning, packet sniffing, password brute force detection and access logs
- **Vulnerability, patching, configuration**, AWS includes automatic software patching, configuration of server details such as password expiration length, and vulnerability scans on OS, web apps, and databases.

Customer responsibility

Cloud is a shared environment, therefore both AWS and the customers are responsible for ensuring proper security



- AWS requires customers to
  - Patch VM guest operating system
  - Prevent port scans
  - Change keys periodically
  - Vulnerability testing of apps
  - Others…

Data issue: confidentiality

- Transit between cloud and intranet
  - E.g. use HTTPS
- Possible for simple storage
  - E.g. data in Amazon S3 encrypted with AES-256
- Difficult for data processed by cloud
  - Overhead of searching, indexing etc.
    - E.g., iCloud does not encrypt data on mail server*
  - If encrypted, data decrypted before processing
    - Is it possible to perform computations on encrypted data?^

Data issue: comingled data

- Cloud uses multi-tenancy
  - Data comingled with other users' data
- Application vulnerabilities may allow unauthorised access
  - E.g. Google docs unauthorized sharing, Mar 2009
  - "*identified and fixed a bug which may have caused you to share some of your documents without your knowledge*."

Cloud Best Practices

2. Data Protection

- *Encryption at Rest:* Encrypt all stored data using strong algorithms (AES-256).
- *Encryption in Transit:* Use TLS 1.2+ for all data transmission.
- *Key Management:* Implement proper key lifecycle management using services like AWS KMS, Azure Key Vault.
- *Data Classification:* Categorize data based on sensitivity and apply appropriate controls.
- *Data Loss Prevention (DLP):* Implement DLP tools to prevent unauthorized data exfiltration.
- *Backup and Recovery:* Regular backups with tested recovery procedures; follow the 3-2-1 rule.

3. Network Security

- *Virtual Private Cloud (VPC):* Isolate cloud resources in private networks.
- *Security Groups and NACLs:* Implement granular firewall rules at instance and subnet levels.
- *Network Segmentation:* Separate production, development, and sensitive workloads.
- *DDoS Protection:* Use services like AWS Shield, Azure DDoS Protection.
- *Web Application Firewall (WAF):* Protect web applications from common exploits.

4. Configuration Management

- *Infrastructure as Code (IaC):* Use tools like Terraform, CloudFormation for consistent, version-controlled deployments.
- *Configuration Scanning:* Regularly scan for misconfigurations using tools like AWS Config, Azure Policy.
- *Hardening Guidelines:* Follow CIS Benchmarks and vendor security baselines.
- *Automated Compliance Checks:* Implement automated policy enforcement.

5. Monitoring and Incident Response

- *Centralized Logging:* Aggregate logs from all cloud resources using SIEM solutions.
- *Real-Time Monitoring:* Use services like AWS CloudWatch, Azure Monitor for continuous monitoring.
- *Security Information and Event Management (SIEM):* Correlate events across multiple sources.
- *Anomaly Detection:* Implement behavioural analytics and machine learning for threat detection.
- *Incident Response Plan:* Develop and regularly test cloud-specific incident response procedures.
- *Forensics Readiness:* Maintain proper logging and snapshots for post-incident investigation.

6. Compliance and Governance

- *Compliance Frameworks:* Align with relevant standards (SOC 2, ISO 27001, PCI DSS, HIPAA).
- *Cloud Security Posture Management (CSPM):* Use tools to continuously assess and improve security posture.
- *Policy Enforcement:* Implement organizational policies through technical controls.
- *Regular Audits:* Conduct internal and external security audits.

7. Container and Serverless Security

- *Container Image Scanning:* Scan for vulnerabilities before deployment.
- *Runtime Security:* Monitor container behavior for anomalies.
- *Function Security:* Apply least privilege to serverless functions, limit execution timeouts.
- *Secrets Management:* Never hardcode credentials; use secrets management services.

Cloud Framework and tools

- **Cloud Security Alliance (CSA)**
  - Cloud Controls Matrix (CCM)
  - Security Guidance for Critical Areas of Focus
  - STAR (Security, Trust, Assurance, and Risk) Registry
- **NIST Cloud Computing Security Framework**
  - SP 800-144: Guidelines on Security and Privacy in Public Cloud Computing
  - SP 800-145: The NIST Definition of Cloud Computing
  - SP 800-146: Cloud Computing Synopsis and Recommendations
- Cloud Service Provider Native Tools:
  - AWS: GuardDuty, Security Hub, Inspector, CloudTrail
  - Azure: Security Center, Sentinel, Azure Defender
  - GCP: Security Command Center, Cloud Armor
- Third-Party Tools:
  - CSPM: Prisma Cloud, CloudGuard, Dome9
  - CASB: Netskope, McAfee MVISION, Zscaler
  - Vulnerability Scanning: Qualys, Tenable
  - Configuration Management: Chef, Puppet, Ansible

IoT and Cloud Security

Modern IoT devices often rely heavily on cloud infrastructure for:

- Data storage
- Device managment
- Machine learning/Ai processing
- Application/Api hosting
  This combination leads to a complex security landscape

- **B. Combined Security Challenges**
  - **1. Extended Attack Surface** The integration multiplies potential vulnerabilities:
    - Device-level exploits leading to cloud compromise
    - Cloud account takeover enabling IoT device control
    - Supply chain attacks affecting both layers
  - **2. Data Flow Security** Data travels through multiple points:
    - Device → Edge Gateway → Cloud → Application
    - Each hop requires appropriate security controls
  - **3. Scale and Complexity** Managing millions of IoT devices connected to cloud infrastructure requires:
    - Automated security orchestration
    - Zero-trust architecture
    - Continuous security validation

- **C. Integrated Security Strategies**
  - **Zero Trust Architecture**
    - Never trust, always verify
    - Micro-segmentation of networks
    - Continuous authentication and authorization
    - Assume breach mentality
  - **Security Orchestration and Automation**
    - Automated threat detection and response
    - Security information sharing between IoT and cloud layers
    - Orchestrated patch management across the ecosystem
  - **Edge Computing Security**
    - Process sensitive data closer to the source
    - Reduce cloud data transmission and associated risks
    - Implement edge-based security analytics

Case Studies

Case Study 1: Mirai Botnet (2016)

- The Mirai malware infected IoT devices using default credentials, creating a massive botnet that launched devastating DDoS attacks. This highlighted the critical importance of secure default configurations and update mechanisms.

- **Lessons Learned:**
  - Change default passwords immediately
  - Implement automatic security updates
  - Network segmentation to limit lateral movement

Case Study 2: Capital One Data Breach (2019)

- A misconfigured AWS web application firewall allowed an attacker to access 100 million customer records. The breach exploited a server-side request forgery vulnerability combined with overly permissive IAM roles.

- **Lessons Learned:**
  - Properly configure cloud security groups
  - Apply principle of least privilege to IAM roles
  - Regular security audits of cloud configurations
  - Implement automated configuration scanning

Conclusion

Conclusions and Key Takeaways

- **IoT and cloud technologies are foundational to modern digital infrastructure**, but they introduce complex security challenges that require specialized knowledge and approaches.
- **Security is a shared responsibility** across device manufacturers, cloud service providers, developers, and end users.
- **Defence in depth is critical**: No single security measure is sufficient; multiple layers of security controls are necessary.
- **The threat landscape is constantly evolving**: Continuous learning, monitoring, and adaptation are essential for maintaining security.
- **Security must be integrated from design through deployment and operation**, not treated as an afterthought.