Cloud computing

- Cloud computing = On-demand access to IT capabilities
- Before cloud computing
  - Everything was stored and processed locally (on-premises)
  - Centralized, secluded and segmented
- Now third party (cloud provider) stores the data.

NIST definition of cloud computing

- 🤓 See also NIST definition of cloud computing

Essential characteristics

- **On-demand self-service**
  - Consumers can unilaterally provision computing capabilities
    - E.g. computing power, storage, network
  - Does not require human interaction
- **Broad network access**
  - Capabilities are available and accessible over the network.
  - Via wide variety of platforms e.g. laptops, mobile phones and PDAs
- **Resource pooling**
  - Uses multi-tenant model to provide resources pooled to serve multiple consumers
    - The instances (tenants) are logically isolated, but physically integrated.
    - One or multiple instances of one or multiple applications operate in a shared environment.
  - Assigns different physical and virtual resources dynamically
  - Location is abstracted to e.g. country, state or data-center
    - Exact location is unknown to user
- **Rapid elasticity**
  - Feeling of "infinite" and instant scalability
  - Usually automatically
- **Measured service**
  - Metering capability in an abstracting way e.g. storage, processing, bandwidth, active user accounts.
  - Resource usage can be monitored, controlled, and reported

Cloud computing service models

Infrastructure as a services (IaaS)

- Capability for consumers to provision processing, storage, networks, and other fundamental computing resources
- Aims to give most control over the provided hardware that runs applications
  - E.g. operating systems, storage, and networking components
- E.g. virtual machines (EC2 in Amazon), virtual networks.

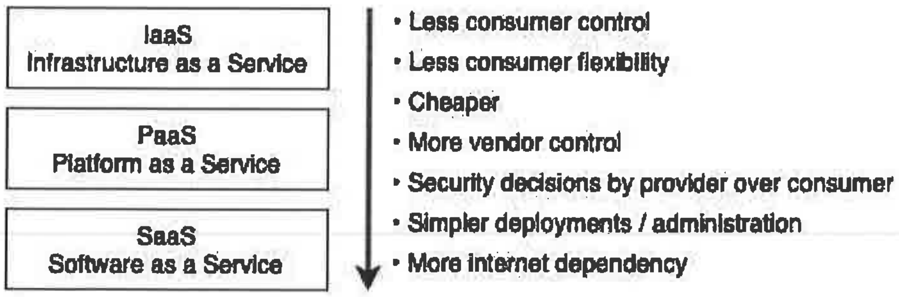
Platform as a Service (PaaS)

- Provides capability for consumers to deploy onto managed cloud infrastructure
- Allows consumer to use programming languages, SDKs, services, and tools supported by the provider
- Consumer does not control or manage underlying cloud infrastructure
  - But can control deployed applications and configurations for the hosting environment
- Provides an environment for building, testing, and deploying software applications
  - Can add features such authentication.
- Aims to help creating an application quickly without managing the underlying infrastructure.
- E.g. development tools, config management, and deployment platforms

Software as a Service (SaaS)

- Software that is centrally hosted and managed for the end customer.
- User does not control underlying cloud architecture
  - E.g. network, servers, OS, storage etc.
  - Can only control limited user-specific application configurations.

IaaS vs PaaS vs SaaS


IaaS
Infrastructure as a Service
PaaS
Platform as a Service
SaaS
Software as a Service
• Less consumer control
• Less consumer flexibility
• Cheaper
• More vendor control
• Security decisions by provider over consumer
• Simpler deployments / administration
• More internet dependency

Identity-as-a-Service (IDaaS)

- Managed authentication services
- Services include e.g. • Single-Sign-On (SSO) • Multi-Factor-Authentication (MFA) • Identity Governance and Administration (IGA) • Access management • Intelligence collection

Security-as-a-Service (SECaaS)

- Integrates security services into corporate infrastructure
- Services include e.g. • penetration testing • authentication • intrusion detection • anti-malware • security and incident management.
- E.g. McAfee Cloud Security

Container-as-a-Service (CaaS)

- Container and cluster services
- Services include e.g. • virtualized container engines • easier container and cluster management
- Inherits features of IaaS and PaaS
- See also container security

Function-as-a-Service (FaaS)

- Provides a platform allowing to develop, run, and manage functionalities without any infrastructure effort.
- E.g. AWS Lambda, Google Cloud Functions, Azure Functions

Separation of duties

- Cloud computing is the ultimate in separation of duties
- E.g.
  - The data owner is the entity accountable for the data itself
  - Data custodian is the entity responsible for access to the data
  - When a single individual becomes both the data owner and the data custodian, security issues can arise.
- Also a countermeasure for insider attacks and social engineering attacks.

Shared responsibility

- **On-premises**: you manage everything.
- **IaaS**: provider manages virtualization, servers, storage and networking
- **PaaS**: provider additionally manages OS, middleware and runtime
- **SaaS**: provider manages everything

Cloud Deployment Models

Private cloud

- Provisioned for exclusive use of single organization
- May be owned/managed/operated by the organization, third party or combination
- May be on or off premises

Public cloud

- No local hardware to manage or keep up-to-date, everything runs on your cloud provider's hardware.
- Services are rendered over a network
- Provisioned for open use by the general public
- May be owned/managed/operated by a business, academic, government or combination.
- Exists on the premises of the cloud provider.

Community Cloud

- Shared infrastructure between several organizations with shared concerns (e.g. compliance)
- Not open to public
- May be owned/managed/operated by the organization, third party or combination
- May be on or off premises

Hybrid cloud

- Composition of two or more cloud (private, community or public)
- Infrastructures remain unique entities
  - but are bound by a technology allowing data and application portability.

Multi cloud

- Multi-cloud is a environment where an organization leverages two or more cloud computing platforms to perform various tasks
- Increases capabilities with combined offering
- Limits data loss and downtime to a greater extent.
- Management products include • Azure Arc • Google Anthos • AWS Outposts

Pros and cons of cloud computing

Advantages of cloud computing

- **Economical**: Less infrastructure cost, less cost of ownership, fewer capital expenses
- **Operational**: cost efficient, elastic, quick provisioning, automatic updates, backup and recovery...
- **Staffing**: Less staff is required, less personal training
- **Security**: Patch application and updates, less cost on security configurations, better disaster recovery, audit and monitoring on providers side, better management of security systems.
- **Innovation**: Quick access to innovation

Disadvantages of cloud computing

- Organizations have limited control and flexibility
- Prone to outages and other technical issues
- Security, privacy, and compliance issues
- Contracts and lock-ins
- Depends on network connections
- Can be hard to migrate from one to another

Cloud regulations

- **FedRAMP**: US regulatory effort regarding cloud computing
- **PCI DSS**: Deals with debit and credit cards, but also has a cloud SIG

NIST Cloud Computing Reference Architecture

- High-level conceptual reference architecture
- Full document

Cloud actors

- **Cloud Consumer**
  - User of the cloud products and services
- **Cloud Provider**
  - Delivers cloud computing based products and services
- **Cloud Auditor**
  - Can conduct independent assessment of cloud services
- **Cloud Broker**
  - Manages the use, performance and delivery of cloud services
  - Negotiates relationships between providers and consumers
  - Service categories
    - **Service Intermediation**: Improves value of a cloud service/function
    - **Service Aggregation**: Combining multiple services to a new one
    - **Service Arbitrage**: Like aggregation but services can be chosen from different vendors.
- **Cloud Carrier**
  - Provides connectivity and transport of cloud services from providers to consumers.

Service Level Agreement (SLA)

- Span across the cloud and are offered by service providers
- Service-level agreement (SLA) is a commitment between a service provider and a client

Cloud security

- Cloud providers implement
  - Limited access and access policies
  - Access logs
  - Ability to require access reason against repudiation

Trusted Computing (TC)

- Attempts to resolve computer security problems through hardware enhancements
- **Roots of Trust (RoT)**: set of functions within TCM that are always trusted by the OS

Cloud computing threats

- **Stealing information from other cloud users**
  - Internal threats where employees copying company data with bad intentions e.g. to trade.
  - Most of those breaches are not published & advertised to media.
  - Information might include e.g. credit numbers, social security numbers
- **Data loss**
  - Deleting data stored on the cloud through viruses and malware
  - ❗High impact if there are no back-ups
- **Attack on sensitive information**
  - Stealing information about other users e.g. financial data.
- **A hacker can utilize computer power** to e.g.
  - crack passwords with many password attempts per seconds
  - DDoS attacks
- **Shadow IT**
  - IT systems or solutions that are developed to handle an issue but aren't taken through proper approval chain
- **Abusing cloud services**
- **Insecure interfaces and APIs** e.g. weak authentication
- **Insufficient due diligence**
  - Moving an application without knowing the security differences
- **Shared technology issues**
  - Multi-tenant environments that don't provide proper isolation
  - If the hypervisor is compromised, all hosts on that hypervisor are as well
- **Unknown risk profile**
  - Subscribers don't know what security provisions are made behind the scenes.
- **Inadequate infrastructure design and planning**
- **Conflicts between client hardening procedures and cloud environment**
- **Malicious insiders**
- **Illegal access to the cloud**

    - E.g. in 2020 United States federal government data breach a compromised global administrator account has assigned credentials to cloud service principals that allowed malicious access to cloud systems.
- **Virtualization level attacks**
- **Privilege escalation via error**
- **Service termination and failure**
- **Hardware failure**: can be mitigated by using more zones in cloud.
- **Natural disasters**: can be mitigated by using more regions in cloud.
- **Weak authentication**
    - E.g. burden of managing identity both on-premises and on cloud
        - Allows compromise on on-premises systems to spread to cloud.
        - Allows adding a malicious certificate trust relationship in cloud for forging SAML tokens on-premises.
- **DDoS** attacks using cloud computing.
- **Compliance risks** e.g. laws regarding data transfer across borders

Cloud computing attacks

- **Social engineering attacks** e.g. password guessing
- **Cross Site Scripting (XSS)**
- **DNS attacks** e.g. DNS poisoning, domain hijacking
- **SQL injection** to to gain unauthorized access to a database.
- **Network sniffing** e.g. obtain credentials, cookies
- **Session hijacking** e.g. cookie stealing
- **Cryptanalysis attacks** e.g. weak encryption
- **DoS (Denial-of-service)**
- E.g. In 2020 United States federal government data breach, used TTP were stealing SAML tokens to attack SSO infrastructure according to TTP analysis from NSA.

Wrapping attack

- Also known as **XML rewriting** attack
- Changes the content of the signed part without invalidating the signature.
- Intercepting a SOAP message and sending/replaying envelope with changed data.

Session riding

- Happens when an attacker steals a user's cookie to use the application in the name of the user
- Simply CSRF in cloud

Side channel attacks

- Also known as • **cross-guest virtual machine breach** • **cross-guest VM breach**
- Attacker controls a VM on same physical host (by compromising one or placing own)
- Attacker can then take advantage of shared resources (processor cache, keys, ...)
- Can be installed by a malicious insider or an impersonated legitimate user

Cloud security tools

- CloudInspect
    - Penetration-testing as a service from Amazon Web Services for EC2 users
- CloudPassage Halo
    - Automates cloud computing security and compliance controls

Containers

- More than one container can run on the same host OS.
- Docker is the standard way of running containers

Containers vs VMs

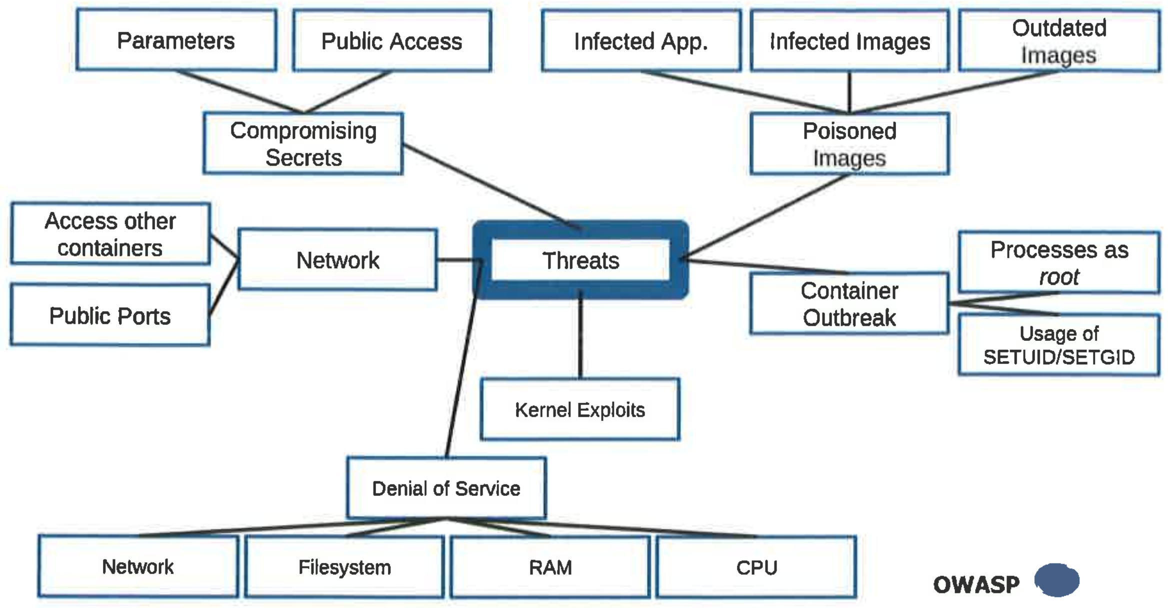
- Both reduces cost due to shared hardware.
  - A single physical server can host multiple, concurrent VMs or applications
- A virtual machine virtualizes an operating system
  - A container sandboxes an applications

Container threats

Container threat model

- **Container escape (system)**
  1. Attacker needs to escape application and end up in container with shell access
  2. Attacker than uses host or kernel exploit to escape container
- **Other containers via network**
  1. Attacker needs to escape application and end up in container with shell access
  2. Attacker attacks other containers on network
- **Attacking the orchestration tool via network**
  1. Attacker needs to escape application and end up in container with shell access
  2. Attacker attacks on management interfaces or other attacks surfaces of the orchestration tool
     - 🤗 In 2018 almost every tool has had a weakness here which was a default open management interface
- **Attacking the host via network**
  1. Attacker needs to escape application and end up in container with shell access
  2. With shell access attacker opens a port from the host
- **Attacking other resources via network**
  1. Attacker needs to escape application and end up in container with shell access
  2. Attacks e.g. on shared network-based file system, LDAP/Active Directory, Jenkins..
     - Can also install sniffer to read traffic of other containers
- **Resource starvation**
  - Attacker causes a container eating up resources which could be CPU cycles, RAM, network or disk-I/O.
  - 📝 Can cause problems affecting all other containers on same host
- **Host compromise**
  - Either through another container or through the network.
- **Integrity of images**

- Each step in CD pipeline where container image is attack vector for the attacker.
- Attacker can inject malicious payloads to deploy.

- 

OWASP Docker Top 10

1. **Secure user mapping**
   - Docker runs with root.
   - Escape from application => root in container
2. **Patch management strategy**
   - Bugs on container/orchestration tools or OS images needs to be patched.
3. **Network segmentation and firewalling**
   - Design your network upfront providing network level protection for
     - management interfaces from the orchestration too
     - network services from the host
   - Expose microservices to only legitimate consumers
4. **Secure defaults and hardening**
   - Ensure no unneeded components are installed or started.
   - Ensure needed components are properly configured and locked down.
5. **Maintain security contexts**
   - Do not mix production containers on one host with other stages of less secure containers.
   - Do not mix frontend with backend services on one host.
6. **Protect secrets**
   - Ensure passwords, tokens, private keys or certificates are protected.
7. **Resource protection**
   - Protect shared resources such as CPU, disks, memory and networks.
   - Ensures one containers usage does not affect others.
8. **Container image integrity and origin**
   - Ensure OS in container is trustworthy until deployment.
   - Ensure images are not tampered with during transfers.
9. **Follow immutable paradigm**

    - Start containers on read-only mode if no file access is needed
10. **Logging**
    - Log on application, container image and orchestration tool
    - Both related events and API level
    - Ensure logs are stored on remote with timestamps and are tamper proof

Container attacks

- Attacks are generally not on the containers themselves, but on the applications running in them.
- Exploiting vulnerable images
    - Container image may have outdated software that has vulnerabilities if it's no longer maintained.
    - E.g. Apache OpenWhisk image (Open Source Serverless Cloud Platform) had a vulnerability where one could replace serverless functions.
- Exploiting bugs and vulnerabilities on unpatched Docker
    - E.g. a bug caused compromised images to access the host OS that has been fixed.
    - E.g. Windows Host Compute Service Shim library had remote code execution vulnerability
    - E.g. a bug allowed root privilige escalation using `docker` command
- Creating malicious container on compromised host system
    - E.g. crypto-miner containers that were running near Russian nuclear warzone
- Exploiting orchestration tool
    - Can be e.g. Kubernetes, OpenShift, Cloud Foundry or other (cloud) layer running containers.
    - See Kubernetes vulnerabilities on CVE

Container advantages over VM

- **Often no SSH enabled into containers**
    - No SSH attacks
- **Often no user access expected**
    - No need for credentials or tools to support users
- **Restricted ports by default**
    - Specific and limiting about which ports to connect
- **Short-lived containers are unlikely bases for attackers**
    - Harder to compromise a service that only lives for a few seconds/minutes.
- **Immutable designs make it difficult to inject malware**
    - As persistance is usually separated away from the container
- **Automatic generation makes it faster to pick up and promote security patches**
    - Automated CI/CD pipelines make updating libraries/OS much quicker than manual
- **Well-defined APIs enables easier anomaly detection**
    - Developers often create APIs to communicate with containers
    - Makes it easy to create a reference model for what is normal inside an application, so anything outside of that is an anomaly. We can automatically detect any anomalies

Container security countermeasures

- OWASP Docker Security Cheat Sheet
    - Keep Host and Docker up to date
    - Do not expose the Docker daemon socket (even to the containers)
    - Set a user
    - Limit capabilities (Grant only specific capabilities, needed by a container)¶
    - Add –no-new-privileges flag
    - Disable inter-container communication (--icc=false)
    - Use Linux Security Module (seccomp, AppArmor, or SELinux)
    - Limit resources (memory, CPU, file descriptors, processes, restarts)
    - Set filesystem and volumes to read-only
    - Use static analysis tools
    - Lint the Dockerfile at build time
- **Pre-deploy sources and dependency validation**
    - Ensures containers are using valid and expected code paths.
- **Pre-deploy authenticity validation**
    - Ensures that the code has not been tampered with.
- **Pre-deploy image scanning for vulnerabilities**
    - Looking for signatures of compromised packages
- **Active vulnerability scans of running containers**
    - Running automated scans after the container is deployed.
- **Network routing that includes traffic inspection**
    - Create a function based service firewall.
- **Integrated log capture**
    - Since there's no local storage, most container patterns are including central log capture and analysis
- **External injection of trust and credentials**
    - Giving credentials just-in-time to running live instances rather than static code
- **Always check running containers**
    - To ensure there's no malicious container running.
- **Ensure container images are up-to-date**
    - Unupdated/stale images might be vulnerable
- **Never-ever as root**
    - Run in rootless mode (as non root)