ST. PÖLTEN UNIVERSITY
OF APPLIED SCIENCES

Informatik & Security

fh st.pölten

Investigation and Automated Detection of Security Threats in Exoscale Cloud

Master thesis

for attainment of the academic degree of

Diplom-Ingenieur/in

submitted by

Benjamin Leopold Baureder, BSc

11942003

in the

University Course Information Security at St. Pölten University of Applied Sciences

Supervision

Advisor: Michael Kirchner, MSc

Assistance: -

Declaration of Honour

First name, Surname: Benjamin Leopold Baureder, BSc

Matriculation number: 11942003

Title of the thesis: Investigation and Automated Detection of Security Threats in Exoscale Cloud

I hereby declare that

- I have written the work at hand on my own without help from others and I have used no other resources and tools than the ones acknowledged
- I have complied with the Standards of good scientific practice in accordance with the St. Pölten UAS' Guidelines for Scientific Work when writing this work.
- I have neither published nor submitted the work at hand to another higher education institution for assessment or in any other form as examination work.

Regarding the use of generative artificial intelligence tools such as chatbots, image generators, programming applications, paraphrasing and translation tools, I declare that

- ☐ no generative artificial intelligence tools were used in the course of this work.
- ☒ I have used generative artificial intelligence tools to proof-read this work.
- ☐ I have used generative artificial intelligence tools to create parts of the content of this work. I certify that I have cited the original source of any generated content. The generative artificial intelligence tools that I used are acknowledged at the respective positions in the text.

Having read and understood the St. Pölten UAS' Guidelines for Scientific Work, I am aware of the consequences of a dishonest declaration.

Kurzfassung

In den letzten Jahren hat sich Cloud Computing in der IT-Landschaft von Unternehmen zunehmend etabliert. Gleichzeitig wurden Vorschriften wie die Datenschutz-Grundverordnung (DSGVO) und der California Consumer Privacy Act (CCPA) eingeführt, die regionale Beschränkungen für die Speicherung und Verarbeitung von Daten auferlegen. Eine Möglichkeit für europäische Unternehmen, diese Anforderungen zu erfüllen, ist die Nutzung eines europäischen Cloud-Service-Providers (CSP). Ein solcher Anbieter ist Exoscale. Da Kunden die Verantwortung für ihre Cloud-Ressourcen mit CSPs teilen, gibt es für die drei größten Anbieter bereits Scan-Tools zur Bewertung der Sicherheitslage von Cloud-Ressourcen. Für kleinere europäische Alternativen gibt es jedoch keine vergleichbaren Lösungen.

Diese Arbeit untersucht potenzielle Bedrohungen innerhalb der Exoscale-Cloud und entwickelt einen Ansatz zu ihrer automatischen Identifizierung mithilfe eines Programms. Zunächst werden die Bedrohungen analysiert, die sich auf die Cloud-Ressourcen der Kunden auswirken können. Anschließend wird ihre Gewichtung anhand der Exposition der Ressource und der potenziellen Auswirkungen auf Vertraulichkeit, Integrität und Verfügbarkeit festgelegt. Die identifizierten Bedrohungen werden anschließend in Kontrollen umgesetzt und in einem praktischen Tool implementiert. Dadurch wird eine systematische und wiederholbare Analyse der Cloud-Ressourcen innerhalb der Exoscale-Cloud möglich. Das daraus resultierende Tool ermöglicht das automatisierte Scannen nach 55 verschiedenen Bedrohungen und Fehlkonfigurationen, wodurch Verbraucher ihre Sicherheitslage verbessern können.

Abstract

In recent years, cloud computing has become increasingly established in corporate IT landscapes. At the same time, regulations such as the General Data Protection Regulation (GDPR) and the California Consumer Privacy Act (CCPA) have been introduced, imposing regional restrictions on data storage and processing. One way for European companies to comply with these requirements is to use a European Cloud Service Provider (CSP). One such provider is Exoscale. Since consumers share responsibility for their cloud resources with CSPs, for the three largest providers, scanning tools to assess the security posture of cloud resources already exist. However, comparable solutions do not exist for smaller European alternatives.

This thesis investigates potential threats within the Exoscale cloud and develops an approach to automatically identify them using a dedicated tool. First, the threats that may affect consumers' cloud resources are analyzed. Their severity is then assessed based on the exposure of the resource and the potential impact on confidentiality, integrity, and availability. The identified threats are subsequently translated into controls and implemented in a practical tool. As a result, a systematic and repeatable analysis of cloud resources within the Exoscale cloud becomes possible. The resulting tool allows automated scanning for 55 types of threats and misconfigurations, enabling consumers to strengthen their security posture.

Contents

1 Introduction

Cloud computing has become a part of modern IT infrastructures, providing an alternative to investments in physical resources and enabling workload-dependent scalability. Driven by emerging trends such as multi-cloud and hybrid strategies, serverless computing, and the integration of machine learning, the global cloud computing market is projected to grow by 21.20% between 2025 and 2030 [1]. At the same time, regulatory frameworks such as the General Data Protection Regulation (GDPR) in the European Union and the California Consumer Privacy Act (CCPA) in the United States of America require organizations to pay close attention to the geographic location of their data storage and processing [2]. Hence, the utilization of a European cloud service provider such as OVH[1], StackIT[2], or Exoscale[3] can facilitate compliance with European regulations.

As cloud service providers operate under a shared responsibility model, consumers remain responsible for the configurations they perform on their resources. This responsibility extends to insecure or incomplete configurations, which can introduce security threats. There are corresponding vulnerability and threat scanning tools for Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP), the three largest cloud service providers [3]. They are either offered directly by the providers, such as AWS Security Hub[4], Microsoft Sentinel[5], Google Cloud Security Command Center[6] or developed and maintained by the community, like Prowler[7]. They allow consumers to check their resources for misconfigurations, vulnerabilities, and related threats.

However, equivalent solutions are currently not available for Exoscale. This gap highlights the need for systematic research into the security implications of configuration options of Exoscale services and the development of mechanisms that help consumers identify insecure practices.

[1] https://www.ovhcloud.com/
[2] https://www.stackit.de/
[3] https://www.exoscale.com/
[4] https://aws.amazon.com/de/security-hub/
[5] https://www.microsoft.com/en-us/security/business/siem-and-xdr/microsoft-sentinel
[6] https://cloud.google.com/security/products/security-command-center
[7] https://prowler.com/

Therefore, this thesis addresses the gap by identifying threats arising from insecure configurations and violations of best practices in Exoscale resources. Based on these threats, corresponding controls are defined and implemented in a prototype that scans resources and reports findings whenever a control is not met. To systematically approach this problem, the following research questions are defined:

- *Which threats emerge from the way consumers can configure and operate resources on the Exoscale cloud platform?*
- *How can the defined threats be detected automatically using a command-line program?*

1.1 Contribution

The contributions of this thesis focus on improving security posture analysis by incorporating platform-specific vulnerability scanning tools. The main contributions can be summarized as follows:

- **Manual Security Analysis**: Cloud resources are systematically evaluated with respect to their configuration options and operating environment. Potential threats are identified and documented, resulting in a structured list of cloud-native threats relevant to Exoscale-hosted resources.
- **Threat Severity Assessment**: Each identified threat is analyzed through a two-dimensional evaluation process that considers the contextual environment of the affected resource. This approach ensures a consistent and reproducible assessment of threat severity.
- **Automated Evaluation Tool**: A prototype tool is developed to operationalize the defined security controls, enabling automated evaluation of Exoscale resources against identified misconfigurations and vulnerabilities.

Collectively, these contributions aim to strengthen the security of Exoscale-hosted resources, assist administrators in detecting and mitigating misconfigurations, and establish a reproducible foundation for future research and tool development in cloud-specific security posture management.

1.2 Thesis Outline

The structure of this thesis is organized to progressively build an understanding of the research context, threat definition, severity evaluation, and prototype implementation. Each chapter con-

tributes to addressing the research question and substantiating the overall argument of the work. The overall structure of this document is as follows:

- **Introduction**: (chapter 1): Introduces and contextualizes the research topic and lays out the motivation for this work. It also outlines the research objectives and questions.
- **Background**: (chapter 2): Provides an introduction to cloud service providers, cloud-native threats, and the cloud service provider Exoscale. Furthermore, a description of cloud services offered by Exoscale, which are addressed in the threat analysis in section chapter 5, is provided.
- **Related Work**: (chapter 3): Reviews existing literature on threats in the context of cloud services and threat scanning tools.
- **Methodology**: (chapter 4): Describes the structured process of defining a threat and assigning a severity value. Also includes the design principles and intended capabilities of the prototype.
- **Implementation**: (chapter 5): Contains the configuration options and the derived threats, taking into account the context and environment in which each configuration is applied. Additionally, it outlines the architecture, core functionalities, and limits of the prototype, highlighting the key functions and their roles.
- **Results and Discussion**: (chapter 6): Presents the threats in the form of controls, which were implemented in the prototype. Revisits the prototype's results and examines the achievement of design objectives, while also discussing the key accomplishments.
- **Conclusion**: (chapter 7): Summarizes the main contributions and findings of the thesis and outlines the future work.

2 Prerequisites

This chapter elaborates on background information on cloud service providers, cloud service models, the shared responsibility model, cloud-native threats, and the cloud service provider Exoscale.

2.1 Introduction to Cloud Service Providers

Cloud computing is defined by the National Institute of Standards and Technology (NIST) as a model that enables network access to a shared pool of configurable computing resources, which can be provisioned and released with minimal management effort. NIST identifies five characteristics: on-demand self-service, resource pooling, broad network access, rapid elasticity, and measured service [4].
NIST also categorized cloud services into the three service models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). These models differ in the degree of control and responsibility assigned to the consumer [4]. Because the division of responsibilities shifts across these models, the concept of the shared responsibility model was established.

2.1.1 Shared Responsibility Model

The shared responsibility model defines which security and operational tasks remain with the provider and which must be ensured by the consumer, making clear that consumers retain accountability for many aspects of security and compliance, particularly configuration decisions. The model differentiates between the three cloud service models and is pictured in Figure 2.1.
In IaaS, the provider delivers fundamental resources such as virtual machines, networks, and storage, while the consumer is responsible for the operating system, applications, and configurations. In PaaS, the provider additionally manages the underlying runtime and middleware, leaving the consumer to focus mainly on application logic and data. In SaaS, the provider delivers a fully

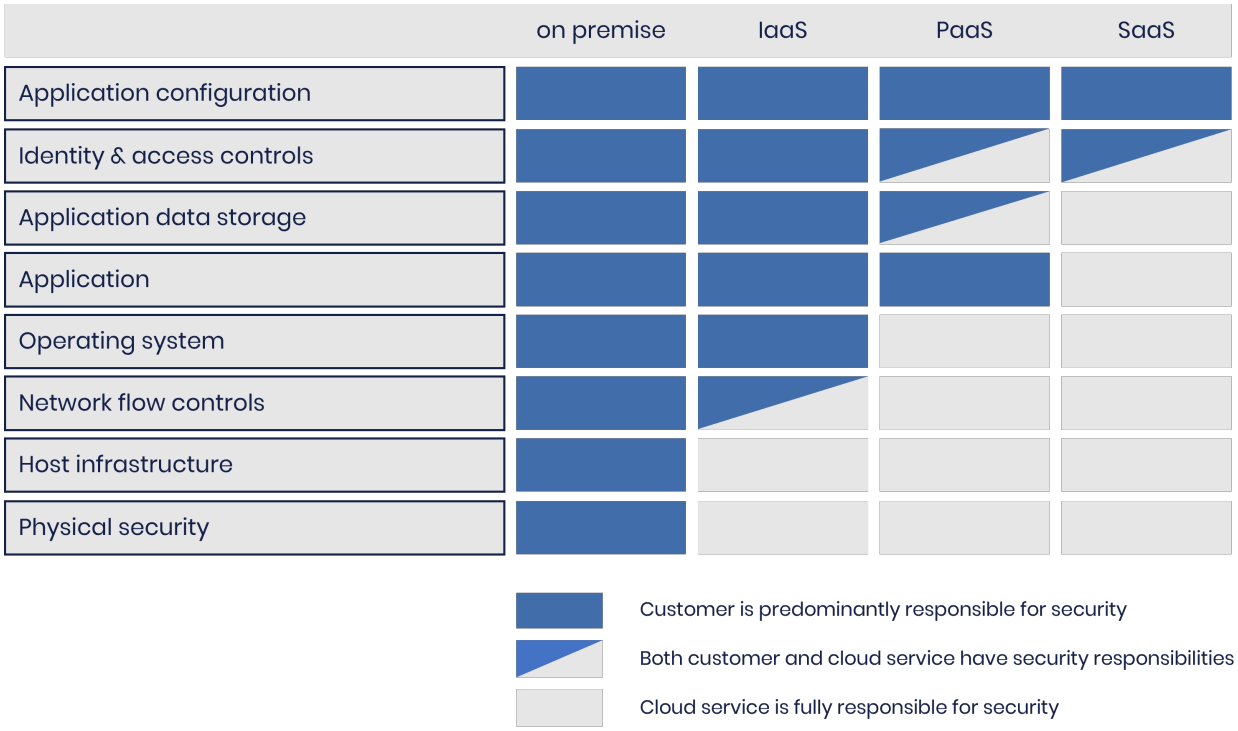

Figure 2.1: An illustration of the shared responsibility model by the British National Cyber Security Center (NCSC). [5].

managed application, and the consumer's responsibility is largely limited to access control and data handling.

Mendoza and Reyes discussed the impact of the shared responsibility model on the cloud security posture and found that neglect and misunderstandings may lead to vulnerabilities [6]. Therefore, this model has a central role in determining the effectiveness of security in cloud environments. By specifying which tasks are the responsibility of the provider and which fall to the consumer, it establishes the basis for secure operation.

2.1.2 Multi-Tenancy

In cloud environments, resources are shared among multiple consumers, which is also referred to as multi-tenancy. Cloud service providers operate large data centers in which physical hardware, such as servers and storage systems, is virtualized by means of hypervisors. This allows several independent virtual machines or services, belonging to different consumers, to run on the same physical host while remaining logically isolated from one another.

The advantages of multi-tenancy include reduced costs through efficient resource utilization, scalable allocation of computing power, and logical separation of tenants to maintain data isolation [7].

2.1.3 Interaction with Cloud Service Provider APIs

Cloud service providers use APIs to enable consumers to access cloud services. The APIs can be used to provision, manage, and share cloud infrastructure. To achieve this, an API client initiates a request for a specific action, which is received by the API endpoint, authenticated, and authorized based on the existing IAM policies [8]. Some providers, such as Exoscale, consolidate multiple component interfaces into a single API endpoint.

2.1.4 Infrastructure as a Service

IaaS allows consumers to provision computing resources such as processing power, storage, and networking for the deployment of arbitrary workloads. While the underlying infrastructure, including physical hardware and hypervisors, is managed by the provider, the consumer is responsible for managing the operating system, storage configuration, backups, and the security of deployed services. Examples of IaaS offerings include AWS EC2, Azure Virtual Machines, Azure Managed Disks, and Google Cloud Virtual Private Cloud (VPC) [4].

2.1.5 Platform as a Service

PaaS provides an environment in which consumers can deploy and manage applications without controlling the underlying infrastructure, such as servers, networks, operating systems, or storage. The responsibility of the consumer lies in managing the application environment itself, including libraries, services, configuration settings, and the application code. Examples of PaaS services include Azure Web Apps, Azure API Apps, and Google App Engine [4].

2.1.6 Software as a Service

SaaS provides the consumer with applications running on a cloud infrastructure, being available to client machines such as a web browser or a program interface. The consumer does not control the underlying infrastructure or program environment and is only responsible for access control and the configuration of the application. Examples of SaaS services include Microsoft Sentinel, Microsoft Office 365, and Google Workspace [4].

2.2 Cloud Native Security Threats

With the adoption of cloud computing, a range of threats emerges that are specific to this environment. The Cloud Security Alliance (CSA)[1] conducted a survey to compile a list of the most relevant cloud-native threats and evaluate them based on importance. In their survey, they specified the subsequent threats [9].

1. **Misconfiguration** Misconfigurations are defined as the incorrect or sub-optimal setup of cloud computing resources, resulting in vulnerabilities, unintended damage, or external as well as internal malicious activity. Some possible misconfigurations are disabled monitoring, insecure automated backups, storage access, lack of validation, unlimited access to ports, and exposed virtual machines [9].
2. **IAM** IAM controls resource access by verifying identities and granting permissions based on user roles. Key components include authentication, authorization, Multi Factor Authentication (MFA), and activity monitoring. Improper implementation or configuration of these features can create security vulnerabilities when unintended permissions are distributed [9].
3. **Insecure Interfaces and APIs** Cloud service providers, enterprise vendors, and internal developers provide APIs and user interfaces for system management. Design decisions made during initial deployment may not be intended for long-term usage, and changes in the environment can introduce new risks. They are vulnerable to issues such as inadequate authentication, lack of encryption, improper session management, missing input validation, poor logging and monitoring, outdated software, and overly permissive access controls [9].
4. **Inadequate Cloud Security Strategy** Cloud security strategy involves the evaluation of external factors, current implementations, and cloud technology choices to develop an approach that supports both security and business objectives. Considerations include cloud architecture, deployment and service models, selection of providers and regions, specific services, and cost models. Strategies may also account for vendor lock-in and regional expansion requirements. Additionally, a cloud security strategy guides the design of identity and access management, networking, and security controls across multiple accounts, providers, and services [9].
5. **Insecure Third-Party Resources** Cloud computing adoption is growing, and third-party resources, such as externally developed code and open-source libraries, introduce potential

[1] https://cloudsecurityalliance.org/

threats. These threats, often considered supply chain vulnerabilities, arise because third-party components are necessary to deliver cloud services or applications [9].

6. **Insecure Software Development** Through the unintended creation of insecure software, weaknesses may be exploited to compromise cloud applications. Cloud service providers offer features such as automated security testing and IAM controls to enforce least privilege and support accountability in development environments [9].
7. **Accidental Data Disclosure** Another threat is the accidental disclosure of data, which can also be caused by misconfigurations. Public search tools can reveal exposed data repositories across platforms such as Amazon S3, Elastic Container Registry, Elastic Block Storage, or Azure Blob Storage [9].
8. **System Vulnerabilities** Another threat is vulnerabilities in systems. These are flaws in cloud services that can be exploited to compromise data confidentiality, integrity, and availability, potentially disrupting service operations. Cloud services can be composed of custom software, third-party libraries, and operating systems, and vulnerabilities in any of these components increase the possibility of cyberattacks [9].
9. **Limited Cloud Observability** Another mentioned threat arises when an organization cannot distinguish between sage or malicious cloud service usage. It can be differentiated between unapproved application usage and misuse of approved applications. Unapproved application use occurs when employees access cloud services without IT authorization, creating shadow IT risks for potentially sensitive data. Misuse of approved applications occurs when organizations cannot track insider actions or external attacks, such as credential theft, Structured Query Language (SQL) injection, or DNS attacks [9].
10. **Unauthenticated Resource Sharing** The unauthenticated sharing of cloud resources poses another threat to cloud resources. This applies to virtual machines, storage buckets, and databases, which could potentially contain sensitive data. With missing user authentication, these resources are exposed to threat actors, who could aim to exfiltrate data. A best practice for securing cloud resources is therefore the implementation of authentication [9].
11. **Advanced Persistent Threats (APT)** Advanced threat actors target cloud resources, employing techniques such as ransomware, extortion, phishing, exploitation of vulnerabilities, and credential theft. To defend against these threats, a multilayered cloud security strategy is essential. Key elements include access controls, encryption, continuous monitoring, and an incident response framework [9].

2.3 Introduction to the Exoscale Platform

Exoscale is a European cloud provider that was founded in 2011 in Lausanne, Switzerland. The platform has grown to offer eight datacenter locations in five countries as of 2025. It is a member of the Austrian A1 Digital International GmbH & Co KG.[2] Apart from DNS, Exoscale offers IaaS, DBaaS, and object storage services. Exoscale services are region-dependent, meaning their availability or configuration options may vary depending on the selected data center. As of July 2025, Exoscale offers the regions displayed in Table 2.1. In the Exoscale documentation, the terms zone and region are used interchangeably, so this practice is also applied in this work.

Data Center	Region
DE-FRA-1	Frankfurt, Germany
DE-MUC-1	Munich, Germany
AT-VIE-1	Vienna, Austria
AT-VIE-2	Vienna, Austria
CH-GVA-2	Geneva, Switzerland
CH-DK-2	Zurich, Switzerland
BG-SOF-1	Sofia, Bulgaria
HR-ZAG-1	Zagreb, Croatia

Table 2.1: Exoscale data centers and their abbreviations [10]

2.3.1 Compute

Exoscale offers a range of compute services that provide virtual machines and container orchestration environments, enabling consumers to deploy applications or host distributed systems. The compute services are based on Kernel-based Virtual Machine (KVM) and run on a shared physical infrastructure. Resources may be restricted to the region in which they were created or be available across all regions, as is the case with Secure Shell (SSH) keys [11].
The Exoscale administrative panel features the following subcategories. They can depend on each other or can influence each other in their configuration.

[2] https://www.a1.digital

Anti-Affinity Groups

Anti-affinity groups allow consumers to influence the resilience of their compute resources, as they ensure that instances sharing a group must spawn on different hypervisors. Once created, the instance stays associated with its respective hypervisor for the duration of its entire existence. Exoscale considers the planning and implementation of a resource distribution strategy to be a best practice [12].

Block Storage

Exoscale provides a redundant and distributed block device implementation for instances. The consumer can create volumes, which serve to host filesystems, and snapshots, which capture the state of a volume at a specific point in time. Block Storage volumes are region-dependent and are available in every region except DE-MUC-1. Data stored in a volume remains in the chosen zone and is replicated twice across multiple nodes. After their creation, volumes can be resized to a larger size. They can be attached to only one instance at a time. Exoscale enforces the limits in Table 2.2 on Block Storage usage [13].

Usage	Limit
Maximum number of volumes attached to an instance	5
Maximum number of snapshots per volume	20
Maximum volume size	10 TiB
Minimum volume size	1 GiB
Maximum read IOPS per volume	5K
Maximum write IOPS per volume	5K

Table 2.2: Exoscale block storage limitations [13]

Elastic IP

Exoscale provides elastic IPs, which can be attached to one or multiple instances in addition to their native IP address. Elastic IPs are available in every region, but, they can only be used within the regions in which they are created. Therefore, the resources to which they are assigned must be located in the same region. When these IP addresses are attached to multiple instances, an even load distribution across instances is not guaranteed. Exoscale offers managed and manual elastic

IPs. In their managed form, no additional configuration is required. Traffic routing includes health check capabilities, which ensure that traffic is only forwarded to healthy instances. Instances with attached elastic IPs handle incoming traffic as if it originated from their native IP address, without distinguishing between traffic received via the native IP and the elastic IP. Outgoing traffic also appears to be sent from the native IP address. In contrast, manual elastic IPs do not offer health check capabilities. Internally, instances attached to a manual elastic IP treat incoming traffic as being addressed directly to that elastic IP, and outgoing traffic can also originate from it.
Exoscale also offers elastic Internet Protocol Version 6 (IPv6) addresses. As the IPv6 address is partially derived from the Media Access Control (MAC) address of the instance, it is also intrinsically tied to it. Consequently, destroying an instance results in the permanent release of its public IPv6 address, as the associated MAC address is removed. Therefore, Exoscale provides consumers with an elastic IPv6 prefix with a prefix length of /96. Contrary to its Internet Protocol Version 4 (IPv4) counterpart, an elastic IPv6 prefix contains over four billion addresses. In managed elastic IPv6 prefixes, only the first usable IPv6 address of the prefix is transparently routed to the associated instances [14].

Instance Pools

Exoscale offers instance pools to enable the provisioning of multiple identical instances automatically. In the pool settings, the instances are defined, and the service keeps the desired number running to achieve high availability. According to Exoscale, a higher degree of elasticity can be provided through dynamic scaling, compared to the creation of single instances. Instance pools are tied to the region in which they are created. Any associated instance is also created in the respective region.
Instance pool settings can be altered after creation, which does not affect already running instances. For changes to take effect, the already running instances must be replaced. This can be done manually by deleting the respective instances or automatically by scaling the pool up and back down again. When an instance pool is scaled down, it replaces the oldest instances first.
An instance pool can be created manually or by other services such as Scalable Kubernetes Service (SKS) or network load balancers. When being referenced by a load balancer, instance pools cannot be deleted [15].

Instances

Exoscale compute instances are virtual machines running in Exoscale datacenters of the respective regions. They are self-contained environments and can run various operating systems. Exoscale offers multiple instance types. An instance type defines the resource configuration of an instance and specifies the number of virtual CPUs, the amount of RAM, and the size of the local SSD storage. In all available regions, CPU-optimized, memory-optimized, and storage-optimized instance types are offered. The regions AT-VIE-1, DE-FRA-1, CH-GVA-2, and AT-VIE-2 also offer access to GPUs. Instances are confined to the zone they are created in and cannot be migrated to another zone post-creation [11].

Load Balancer

Exoscale offers layer four load balancers to distribute ingress traffic to the compute instances of an instance pool. One load balancer may be configured to serve multiple services. Each service is bound to an instance pool, which must be in the same zone as the load balancer. Therefore, load balancers are region-dependent. They are available in every region. Load balancers expose a single IP address for all services. They offer three types of balancing strategies. Round-robin forwards incoming traffic to each instance pool member in equal proportions and in circular order. Source-hash balances incoming traffic on each instance pool member depending on the source IP address. The Maglev-hash forwards each destination an approximately equal number of connections. Instances remain accessible individually through their public IP. Through automatic updates, the load balancer ensures that every available instance within an up- or downscaled instance pool is considered as a target. Services also include a health check functionality to exclude unreachable instances. Outgoing traffic is sent directly from the pool member instance.
Every load balancer can contain up to ten services, each independent and with its own individual configuration. They are incapable of terminating encrypted connections, as they only forward traffic to pool members. Load balancers are subject to the default limit of five per account, and their exposed IP counts towards the elastic IP limit as well. It is not supported to provide an arbitrary list of instances, as load balancer services can only be associated with instance pools [16].

Private Networks

Exoscale allows consumers to create private networks. Instances can be provisioned with up to eight private network interfaces to communicate with other instances within the same network. Consumers can choose to create a managed network, in which a server provides Dynamic Host Configuration Protocol (DHCP) services, or a manual network, where IP configuration is handled by the consumer. Private networks are local to a region and are available in every offered zone. Security group rules are not applied to traffic inside private networks, and data transfer is not encrypted. The largest supported MTU size is 1500. Therefore, jumbo frames are not supported [17].

Security Groups

With security groups, Exoscale provides the ability to define firewall rules for compute instances. These rules control both incoming and outgoing network traffic and are enforced at the hypervisor level. Security groups can deliver isolation similar to a Virtual Local Area Network (VLAN) while maintaining a single public IP.
A security group can contain up to 60 rules, all of which override the default network behavior that allows all outgoing traffic and denies all incoming traffic. Once a custom outbound rule is defined, only traffic matching the specified outbound rules is permitted. Instances associated with a security group can also be referenced as traffic sources or destinations in rules.
Exoscale additionally provides Open Systems Interconnection (OSI) layer two filtering capabilities, which are automatically applied. These mechanisms prevent MAC address spoofing, Address Resolution Protocol (ARP) spoofing, DHCP server spoofing, and packet snooping of neighboring instances through capture tools. At layers three and four, ingress and egress traffic can be filtered by protocol, destination IP, and destination port. To prevent unknown unicast, broadcast, and multicast traffic from leaving an instance, egress filtering is applied. Applications that rely on broadcast, multicast, or unknown unicast traffic are therefore not supported on Exoscale [18].

SKS

Exoscale offers a managed Kubernetes service that is integrated with the rest of its ecosystem. The service provides a managed control plane and supports scaling by adding or removing node pools. Node pools consist of instance pools, while nodes themselves are individual instances running Kubernetes workloads. They are configured using cloud-init scripts. Automatic upgrades

to the latest Kubernetes patch versions are supported. SKS clusters are region-dependent resources and are available in every Exoscale region.
The control plane, including the API server, Controller Manager, Scheduler, and Konnectivity server, is operated entirely by Exoscale. Consumers have no access to the control plane. The Konnectivity server provides a proxy to facilitate secure communication between the control plane and cluster. Core components automatically deployed in each cluster include Kube Proxy, CoreDNS, the Konnectivity agent, Calico, and the Metrics Server. Kube Proxy manages traffic forwarding to pods in iptables mode. CoreDNS provides internal cluster DNS and, once deployed, is not updated by Exoscale. For networking, Exoscale offers either the Container Network Interface (CNI) Calico or Cilium.
Node pools are logical groups of Kubernetes worker nodes, supplied by instance pools. Consequently, the same configuration settings apply as described for instance pools and instances. A cluster can contain multiple node pools. However, once created, the instance pool associated with a node pool cannot be managed directly. Operations such as scaling or replacing nodes must be performed at the node pool level.
Exoscale defines two best practices for ensuring cluster stability. First, workloads should be configured with appropriate resource requests and limits to avoid contention. Second, the health status of nodes should be continuously monitored to ensure reliable operation.
Certain limitations apply to SKS clusters. Volumes can only be attached to one node at a time, and online resizing of PersistentVolumes is not supported. Root API credentials automatically expire after 30 days and must then be rotated or renewed to maintain access [19].

SSH Keys

Exoscale supports the use of SSH key pairs as an authentication method for Linux instances. During provisioning, an instance can be initialized with a single key, while additional keys may be added after creation. The supported SSH key formats are `ssh-rsa` and `ssh-ed25519`. SSH keys are managed as a global resource and can be used across all regions [20].

Templates

Exoscale allows consumers to create custom templates in addition to the provided default templates. This feature enables the deployment of custom operating systems or tailored template

configurations. The image must be in a KVM compatible format or it will fail to launch. The supported virtual disk size ranges from 10 GB to 1 TB. To ensure full compatibility with the cloud platform, it is recommended by Exoscale to install cloud-init on the custom image. Custom templates are local to the region in which they are created. To use a template across multiple regions, it must be recreated separately in each region [21].

2.3.2 Object Storage

Exoscale offers object storage in the form of buckets, which serve as containers for storing a potentially unlimited number of objects. An object consists of the data itself, associated metadata, and a unique identifier within the bucket. This identifier also functions as the object's name. The metadata is a set of name–value pairs that configure or describe the object, and it can either include standard Hypertext Transfer Protocol (HTTP) headers or be customized by the consumer.
To meet its Service Level Agreement (SLA) of 99.95% availability, Exoscale replicates stored data across three independent nodes. Objects can also be replicated to one or multiple buckets across zones, supporting disaster recovery, backups, or multi-region synchronization. All data and replicas remain stored within the country of the chosen zone, while being accessible across all regions offered by Exoscale.
Exoscale further provides features to manage and secure access. Access Control Lists (ACL) can be used to define permissions and control access to objects, while Cross-Origin Resource Sharing (CORS) allows the configuration of cross-origin headers, enabling browser-based applications and websites to interact with object storage. Additionally, object lock can be configured for retention and archiving purposes, and versioning is available to preserve multiple iterations of the same object [22].
Encryption in transit is ensured through the use of Hypertext Transfer Protocol Secure (HTTPs). Encryption at rest, however, falls within the consumer's responsibility. This can be achieved either by encrypting objects before uploading them to Simple Object Storage (SOS) or by using Server-Side Encryption by Customer-Provided Keys (SSE-C), which, as of August 2025, is not supported [23].

2.3.3 DBaaS

Exoscale provides managed data stores that can be deployed automatically and are available in every region. All data is stored within Exoscale's regions, located in Europe. By default, DBaaS resources are not publicly accessible from the internet, and data is never transmitted unencrypted, as Secure Sockets Layer (SSL) encryption and authentication are enabled. However, these settings can be changed by the consumer. The service is covered by Exoscale's SLA, which guarantees 95.95% uptime for the Hobbyist and Startup service plans, and 99.99% uptime for the Business and Premium plans [24].

MySQL

Exoscale offers a managed MySQL 8 service, as well as MySQL 5.7 for legacy workloads. The platform handles provisioning, patching, scaling, and backups. Database instances are local to their respective regions but are available in every Exoscale region. Depending on the selected service plan, consumers can choose from various scaling options, automatic backups, and point-in-time recovery. To offload read traffic, read replicas can be deployed. While primary clusters operate within a single zone, geo-redundancy can be achieved through the integration of read replicas. Access to super user rights is restricted and must be requested by consumers via Exoscale support [25].

PostgreSQL

Exoscale offers a managed PostgreSQL service. The platform manages provisioning, patching, backups, and minor version upgrades, reducing operational overhead for users. PostgreSQL instances are local to their respective regions but available in all Exoscale regions. Depending on the chosen service plan, users can take advantage of scaling options, automatic backups, point-in-time recovery, and read replicas to offload read traffic. Additionally, with pgBouncer, a connection pooler fronting every service is used for a higher connection density. Primary clusters operate within a single zone, while geo-redundancy can be achieved using integrated read replicas. Access to super user rights is restricted and must be requested by consumers via Exoscale support [26].

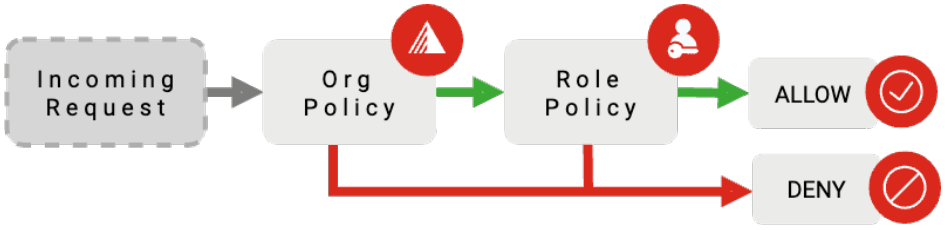

Figure 2.2: Authentication flow in Exoscale [28]

2.3.4 IAM

With IAM, policy-based permissions can be defined to regulate the actions a user or associated API key is permitted to perform. In the audit trail, every action performed can be retraced [27].
Exoscale employs a split authorization model consisting of an organizational policy and a role policy. The organizational policy is evaluated first and applies universally to all users and API keys. The role policy is applied only if the organizational policy allows the requested action, and it is assigned to specific users or API keys. For a request to be successfully executed, it must be authorized under both the organizational and the role policy [28]. The authentication flow is depicted in Figure 2.2.

Users and Keys

Any action performed on the cloud platform is done in the context of an organization. They represent a workspace for multiple users. Users of an organization are individuals with access to the web portal, enabling them to control or monitor the cloud infrastructure and platform through the user interface. The user who creates the organization is designated as its owner and is granted full permissions within the organizational context. Additional users may be assigned the owner role, but at least one owner must remain assigned at all times. Another default role is the billing role, which allows exclusively the management of administrative and billing information.
API keys can also be associated with roles. However, contrary to users, it is impossible to modify an IAM role after creation [29].

Roles and Policies

Exoscale provides multiple interaction methods with its platform, but they are all subject to the permissions set in IAM. IAM features roles, which act as a container for a single policy, and policies,

which are a set of rules describing the permissions. Roles can be assigned to users and API keys. A policy consists of a default service strategy, either allowing or denying all actions for resources that are not specified in the services map, which contains the services and respective allowed or denied actions [28].

2.3.5 DNS

Exoscale provides functionality to host DNS zones and manage their records. However, it does not offer domain registration services. Domains must be registered through external registrars, and the DNS servers of the respective zone need to be adjusted with the registrar accordingly [30].

3 Related Work

This section provides an overview of the literature relevant to this thesis, focusing on works that address related topics or contribute insights. The reviewed sources include implementations and analyses of used scanning tools, evaluations of threats and misconfigurations in cloud environments, as well as security assessments of cloud service providers.

Early work by Cao and Liu (2016) introduced a security scanner for private cloud solutions, targeting vulnerabilities, viruses, and malicious behavior via the hypervisor. The authors discussed cloud-specific characteristics for vulnerability scanning, such as scalability and support for an unspecified number of resources. The paper presents an approach that leverages the privileged position of the hypervisor to perform vulnerability, virus, and malicious behavior scanning [31].

To strengthen cloud infrastructure security, Jimmy Fnu (2023) identified Cloud Security Posture Management (CSPM) as a key approach, emphasizing continuous assessment of security configurations. Leveraging big data and AI trends, the author proposed a CSPM tool for AWS based on the NIST CSF v1.1 framework [32].

Yasani et al. (2024) examine future trends in intelligent threat detection and mitigation for cloud infrastructures. Recognizing the dynamic nature of the environment and evolving cyber threats, they highlight the advantages and possible advancements of artificial intelligence in cloud security [33].

A security assessment was conducted by Gordin et al. (2021) of the private cloud solution OpenStack[1], using the vulnerability scanning tools Nessus[2], Metasploit Pro[3], and OpenVAS.[4] This study conducted security assessments inside and outside the cloud network environment. The evaluated scanning tools were compared based on their results, with Nessus identified as the most comprehensive solution [34].

[1] https://www.openstack.org/

[2] https://www.tenable.com/products/nessus

[3] https://www.metasploit.com/

[4] https://www.openvas.org/

One of the earlier works on automated cloud security compliance tools was presented by Ullah et al. (2013). The authors combined several approaches to gather information from OpenStack, including API access, vulnerability scanning, and log analysis. Their identification of challenges in automating compliance checks is particularly relevant for developing a prototype tailored to Exoscale [35].

Mejri et al. (2021) analyzed cloud security issues and proposed a log-based proactive strategy. They emphasized the importance of configuration analysis to prevent compromises of cloud resources and employed tools such as Checkov[5], Terrafirma[6], CloudSploit[7], and general vulnerability scanners. However, none of these tools support the Exoscale platform, highlighting the need for a dedicated solution [36].

Goel and Singhal (2023) discussed security issues and threats specific to cloud computing. The authors acknowledge current trends and the importance of cloud security, while also discussing the issues regarding privacy, accuracy, and accessibility of consumer data and services. They recognized data security and privacy as the main concerns, formulated threats, and proposed measures to improve cloud computing security [37].

Mary et al. (2023) provided an analysis of security challenges and threats in cloud environments. The authors evaluated contemporary security standards, listed cloud-native security challenges, and proposed mitigation measures [38].

Narayanasamy (2025) discusses the shared responsibility model and measures to improve the security posture in cloud environments. The author recognizes best practices and appropriate security measures. They focused on the identification of preventative cloud security measures [39].

This research supports the identification of vulnerabilities in cloud environments that arise from resource configurations and underlines the need for threat-scanning tools capable of interacting with the API of cloud service providers. It further emphasizes the importance of expanding existing tools to include additional providers, such as Exoscale.

[5] https://www.checkov.io/

[6] https://github.com/wayfair/terrafirma

[7] https://github.com/aquasecurity/cloudsploit

4 Methodology

As stated in chapter 2, cloud service providers employ a shared responsibility model, according to which the consumer retains responsibility for the configurations they are able to modify. The possible configurations include, but are not limited to, the software and operating systems used, adopted default configurations, and public exposition of resources. Configurations also may contradict existing best-practice guidelines. The configuration options have the potential to pose threats to the confidentiality, integrity, or availability of consumer resources in the cloud.
This research employs a qualitative security assessment, focusing on the manual configuration analysis of the Exoscale cloud platform. This serves to identify potential threats to cloud resources that may arise from misconfiguration or insecure default settings. In addition, a prototype is developed to detect the identified threats in a program-controlled manner.

4.1 Definition of Analysis Scope: Exoscale Service Categories

The cloud service provider Exoscale offers a broad range of services, each with distinct configuration options that remain the responsibility of the consumer. As of July 2025, the administration portal organizes these services into the following categories: *Compute*, *Storage*, *DBaaS*, *DNS*, *IAM*, *Marketplace*, *Organization*, and *Support*. Within each service category, Exoscale further distinguishes subservices, hereafter referred to as subcategories [40]. Not all categories were included in the analysis, as some either fall outside the scope of base-cloud services, lie outside the consumer's responsibility under the shared responsibility model, or do not expose configurable options to the consumer.
The services in the category *Compute* allow consumers to manage their instances, instance pools, SKS, block storage, security groups, load balancers, private networks, SSH keys, and anti-affinity groups. It is split into the corresponding subcategories. Consumers can request resources and configure them according to their application case. Since resources in this category can be con-

figured directly by consumers, and the responsibility for secure configuration lies with them, it is therefore included in the manual analysis [41].

The category *Storage* offers SOS, enabling users to create object storage buckets, share their data, and manage access permissions. Therefore, this category is included in the manual analysis [42].

In the *DBaaS* category, users can launch their database instances. Supported are the SQL databases MySQL and PostgreSQL. In addition, Exoscale lists the services Apache Kafka, OpenSearch, Valkey, and Grafana under the category NoSQL databases and Database Management Systems (DBMS). While Kafka is technically a data streaming platform, OpenSearch is an indexing and search tool, and Valkey is a key–value store; they are grouped by Exoscale in this category and are therefore referenced as such in this work. The category is split into subcategories according to the supported services. Furthermore, the configuration options include access rules and other database-specific settings [43].

The *DBaaS* category contains services that can be publicly exposed and where consumers are responsible for configuring access and related settings. However, according to a survey by Stack Overflow[1], PostgreSQL was the most widely used database in 2024, followed by MySQL [44]. To reflect this usage and to provide a proof of concept, the manual analysis in this work is limited to the two SQL-based services.

In the *DNS* category, Exoscale provides users with the ability to manage their DNS domains and create DNS records. However, since this category does not involve any cloud-specific configuration options but rather standard DNS configuration capabilities, it is not included in the manual analysis [45].

The *IAM* category allows consumers to create roles with specific permissions, manage and assign users, generate API keys, modify the organizational policy, and configure Single Sign-On (SSO). In addition, it includes references to documentation on how to interact with the API. Since access permissions to other resources can be configured, this category is included in the manual analysis [27].

The Exoscale *Marketplace* allows users to access third-party templates and managed services. This category is not included in the manual analysis because it does not involve native cloud configuration options. Furthermore, it offers third-party solutions that are not part of the core Exoscale cloud platform and therefore fall outside the scope of the analysis [46].

[1] https://stackoverflow.com/

The category *Organization* contains information on accounting, the audit trail, setting application quotas, and compliance documents of the cloud service provider. The *Support* category provides an overview of new, open, and closed support tickets. Since neither category allows security-related or resource-specific configurations, both are omitted from the manual analysis.
The categories included in the manual analysis are therefore *Compute*, *Storage*, *DBaaS*, and *IAM*.

4.2 Manual Analysis and Control Definition

Each selected category, along with its subcategories, undergoes a structured analysis of security-relevant configuration settings and resource-specific conditions. The process consists of an initial identification of configuration options and conditions. Subsequently, the potential threats to the data and infrastructure of consumers are assessed. This is followed by a structured severity classification of each threat.

Security Posture Assessment

The resource is analyzed based on best practices provided by Exoscale or other cloud service providers, institutions with relevant guidelines such as the Federal Office for Information Security (BSI), and tools focused on cloud security posture for other public providers.

Threat Definition and Severity Classification

For each resource type, the configuration options, operational environment, and effect on associated resources are analyzed, and a threat is defined. To classify the severity of a threat, a scoring system is used. This approach is inspired by FIRST's[2] Common Vulnerability Scoring System (CVSS), in which the overall score is composed of four metric groups. In the case of this work, a two-dimensional scoring system is applied. The two dimensions are the potential impact on confidentiality, integrity, and availability of the identified threat and the exposure of the resource. Both are scored on a scale from one to three, with the sum determining the overall severity level. This scoring enables context-sensitive evaluation, as technically identical configurations may lead to different severities depending on the deployment environment. The classification of the potential impact involves assessing whether the issue could lead to unauthorized data access, service dis-

[2] https://www.first.org/

ruption, privilege escalation, or loss of control over critical components. The classification criteria for the potential impact score are shown in Table 4.1.

Score	Potential Impact	Description
1	Low	No effect on confidentiality, integrity, or availability. Best practices are ignored, or cloud resources consume monetary resources without any apparent benefit.
2	Medium	Potential to compromise either confidentiality, integrity, or availability under specific conditions or when combined with other weaknesses.
3	High	Confirmed effect on at least one of confidentiality, integrity, or availability, or potential to affect multiple.

Table 4.1: Potential impact to Confidentiality, Integrity, and Availability (CIA) scoring criteria

The classification of the exposure provides context in the evaluation process, considering that not all resources within a cloud environment are public or equally exposed. The aim of adding this dimension is to take this circumstance into account. The classification criteria for the exposure score are shown in Table 4.2.

Score	Exposure	Description
1	Low	Consumer resources are either not publicly accessible by design or configured to be unreachable.
2	Medium	Consumer resources are publicly accessible with restrictions such as IP filters.
3	High	Resources are user accounts, API keys, or might be exposed to the internet without restrictions.

Table 4.2: Exposure scoring criteria

Based on the addition of scores in the dimensions of potential impact and exposure, the threat receives an overall score, which is used to classify the severity of the threat. The points assigned to the overall severity scale can be seen in the Table 4.3.

Overall Score	Severity Level
2	Informational
3	Low
4	Medium
5	High
6	Critical

Table 4.3: Severity level classification based on total score

4.3 Cloud Security Evaluation with Automated Tooling

To achieve the automated evaluation of the identified threats, a designated tool is needed. Each threat defined during the manual analysis phase is translated into one or more controls. The controls are then integrated into a Python-based prototype.

Control Implementation

Before the implementation of a control, the Exoscale API is analyzed to determine what data about the necessary cloud resources can be retrieved. A feasibility analysis is then conducted based on the data returned by the API. Based on the format and attributes of the API response, the resources are mapped to Python-classes, enabling structured parsing and further analysis.

Design Principles and Intended Capabilities

The prototype is designed to enable an automated, repeatable evaluation of security-relevant cloud configurations. It is designed in a modular structure to enable the integration of additional controls and the support of services that might be added in the future. Evaluation results are printed on the Command-Line Interface (CLI).

5 Approach

This chapter outlines the manual analysis of the Exoscale service categories, which were selected in chapter 4. Threats arising from certain consumer settings are identified and classified. The identified threats are then be translated into controls for the prototype. Also, an outline of the prototype is provided, detailing its key functions and overall functionality.

5.1 Analysis of Exoscale Services

This section explains the configuration options of resources within each category and subcategory selected in the methodology. If the configuration is deemed security-relevant, the process continues with the definition of the associated threat. To improve readability, the potential impact on confidentiality, integrity, and availability is referred to as the potential impact. The exposure level is set for each resource type, whereas the potential impact is evaluated for each threat. If a specific threat has a different exposure level, the reasoning is provided in the description of that threat, where it is addressed separately.

5.1.1 Compute

This subsection analyzes the configuration options of the *Compute* subcategories, outlines their resource-specific exposure levels, and presents the identified threats together with their assigned severity ratings.

Anti-Affinity Groups

Resource Configuration Options: Upon the creation of anti-affinity groups, the consumer can set the name of the group and provide a description. Neither of those options has a potential impact on the group's security.

Exposure Level: For anti-affinity groups, the exposure level is generally low, as they cannot be made publicly accessible.

Possible Threats: The omission of defining a resource distribution strategy with anti-affinity groups may result in reduced resilience of the cloud infrastructure compared to a deployment implementing such a strategy. A threat assessment is conducted with the respective resources, as such an evaluation requires the consideration of the specific resources utilizing anti-affinity groups.

Block Storage

Resource Configuration Options: Upon the creation of block storage volumes, the consumer can set the name of the volume, choose the zone in which the data is stored, and specify the size of the volume. Volumes can be created blank or based on a snapshot. When creating a snapshot of a volume, the consumer can set the name of the snapshot. Block storage encryption settings cannot be specified.

Exposure Level: For block storage resources, the exposure level is generally low, as neither volumes nor snapshots can be made publicly accessible.

Possible Threats:

- *Missing automatic snapshots:* Snapshots are not created automatically and are hence the responsibility of the consumer. Therefore, the consumer is also responsible for data backup plans. Failure to do so may result in data loss if the volume is intentionally or unintentionally deleted or corrupted. This results in a high potential impact level, as it has the potential to compromise the availability of consumer data. Together with the low exposure level, this results in an overall medium severity level.
- *Unattached volumes:* Another possible configuration constitutes the presence of unattached volumes without associated snapshots. This represents additional financial expense without benefit to the consumer. Such volumes generate costs despite being unused. The intention of using unattached volumes as a form of data retention contradicts Exoscale's best practices, as the provider explicitly recommends the use of snapshots for this purpose. The presence of unattached volumes without snapshots not only results in potentially unnecessary expenditure of the consumer's budget but could also indicate the absence of a

data-retention strategy. In some cases, such volumes might be intended for data retention, but without snapshots, this approach infringes on the best practices set by Exoscale. Given these circumstances and the potential impact on data availability, the potential impact level is assessed as high. Together with the low exposure level, this results in an overall medium severity level.

Elastic IP

Resource Configuration Options: When creating an elastic IP, the target zone in which the resource will reside must be specified, along with the IP version (IPv4 or IPv6) and whether it will be managed or manual. An optional description can be provided, and for managed elastic IPs, health check parameters can be configured, supporting either a general TCP mode or HTTP/HTTPS. When specifying health check parameters, Exoscale allows the selection of a port, the check interval in seconds, a timeout, and the number of consecutive failures required before the resource is considered unhealthy. Each parameter is subject to limits defined by Exoscale.

Exposure Level: For elastic IPs, the exposure level is generally high, as they are always public. This can be overridden by an elastic IP not being attached to resources or having unsuccessful health checks, which results in traffic being dropped.

Possible Threats:

- *Failed health checks:* If a health check fails, traffic is no longer forwarded to the associated instance. The elastic IP remains active and continues to incur costs, while the underlying service becomes unavailable. This affects the availability of the workloads, resulting in a high potential impact level. Although the elastic IP is public, no traffic is forwarded to any resources when the health check fails, so the exposure level is considered low. Combining the medium potential impact level with the low exposure yields an overall score of medium.
- *Unattached elastic IPs:* As with block storage volumes, unattached elastic IPs continue to incur costs while not providing immediate benefits, as they are not attached to any instance and therefore remain unused. However, consumers might want to retain an IP address because other parts of their infrastructure are tied to it, or because it has not yet been indexed by services like Shodan.[1] Therefore, the assessed potential impact level is low.

[1] https://www.shodan.io/

Although elastic IPs are always public, when they are not attached to other resources, they do not expose the resources of a consumer, which also results in a low exposure level. This results in the lowest possible overall score, which is informational.

- *Associated entries from indexing services:* Also worth considering are entries of indexing services, such as Shodan, associated with an elastic IP. These entries can reveal potential vulnerabilities in the infrastructure of a consumer, such as Common Vulnerabilities and Exposures (CVE) records tied to the elastic IP in use. However, because elastic IPs can be requested and released on demand, previous owners may have operated their infrastructure insecurely or even run honeypots, causing the IP to appear in Shodan with associated records of CVEs. This increased visibility can result in heightened attention from attackers toward that IP. As this can have a potential impact on the availability, confidentiality, and integrity of resources associated with the elastic IP, the potential impact level is deemed high. Together with the high exposure level, this results in an overall critical severity level.

Instance Pools

Resource Configuration Options: When creating an instance pool, the pool name and size must be set. Optionally, the prefix of member instances and a description can be configured. The pool size is limited by a default quota of four instances per pool. The default quota can be increased by requesting a quota increase. Furthermore, additional configuration options that define instance parameters need to be set. This includes the zone, template, and member instance size. Since these options fall under the subcategory *Instances*, they are examined in greater detail within that subcategory. These configuration options can be altered when the instance pool is in a running state.

Exposure Level: Instance pool member instances may be restricted by security groups, but they can still be publicly exposed and potentially targeted by attacks intended to disrupt service, such as a Denial of Service (DoS). Through the possible restrictions, the exposure level is medium.

Possible Threats:

- *Member instances not in a running state:* When utilizing instance pools, an intended goal can be the distribution of identical workloads across multiple instances to increase capacity and provide fault tolerance. However, this cannot be achieved if member instances exist but

are not in the running life-cycle state. In such cases, the actual available capacity is lower than the planned capacity, which can result in workloads not operating as expected. This affects the availability of services, which is reflected in a high potential impact level. Together with the medium exposure level, this results in an overall high severity level.

- *Different instance types in pool:* Another threat that arises when the intended goal is to distribute identical workloads across multiple instances is the operation of non-identical instances within the same instance pool. Through modification of instance settings, the member instance types may no longer match, and smaller instances may fail to handle the workload as expected. This can lead to degraded service performance, such as increased latency or even outages. This warrants a high potential impact level. Together with the medium exposure level, this results in an overall high severity level.
- *Different instance templates in pool:* Another threat stems from updating the instance pool settings after creation. In such cases, the instance template may no longer be consistent across all member instances. This can result in diverging environments for the service running on those instances, ultimately preventing the service from being delivered as intended. Since this directly affects availability, the potential impact level is assessed as high. Together with the medium exposure level, this results in an overall high severity level.

Instances

Resource Configuration Options: When creating an instance, the template, instance type, zone, and disk size must be set. Optionally, the instance name, SSH keys, private networks, security groups, and anti-affinity groups can be chosen. It is also possible to assign a public IPv4 address, a public IPv4 and public IPv6 address, or no public address. Exoscale instances can also enable the Trusted Platform Module (TPM) or Secure Boot. To make further configuration changes, user-data scripts can be supplied. User-data scripts allow configurations of the virtual machine at first startup of the machine and can be used to interact with the OS. Since the options security groups, anti-affinity groups, and private networks fall under different subcategories, they are examined in greater detail within their respective subcategory.

Once an instance is created, it can be configured to support IPv6 and to enable snapshot functionality. Snapshots of instances differ from block storage volume snapshots, as they are linked to the instance and are deleted upon the deletion of the instance.

Exposure Level: Instances may be restricted by security groups, but they can still be publicly exposed and potentially targeted by attacks intended to disrupt service, such as a DoS. Through the possible restrictions, the exposure level is medium.

Possible Threats:

- *Instances without snapshots:* Instances that are in a running state and already providing their assigned workload should be snapshotted, as this can be part of a disaster recovery plan. If no snapshot exists, data and services may not be recoverable in the event of an instance failure. This is primarily the case with stateful instances, where files might be stored locally. This represents a potential impact on the availability of data, provided no other compensating measures are in place, which warrants a medium potential impact level. Together with the medium exposure level, this results in an overall medium severity level.
- *Majority of instances in same zone:* Distributing instances across multiple zones increases resilience when compared to concentrating them in a single zone. In case of a zone outage, workloads can continue to operate from other zones, whereas a single-zone deployment may result in a full service outage. The extent to which service outages can be tolerated varies depending on the specific infrastructure, its requirements, and the individual resource loads. For this analysis, a tolerance level of 75% is defined for the associated controls. Not distributing instances directly impacts the availability of services, which leads to a high potential impact level. Together with the medium exposure level, this results in an overall high severity level.
- *Instances sharing a host:* Assigning anti-affinity groups to instances ensures that instances within the same group are not scheduled on the same physical host. If this configuration is not set, a single host failure can affect multiple instances at once. Therefore, the use of these groups serves to improve resilience and thus the availability of data and services. As the potential impact directly concerns the availability dimension, the potential impact level is assessed as high. Together with the medium exposure level, this results in an overall high severity level.
- *Instances running deprecated services:* Another possible threat arises from instances that exceed a certain age threshold. In this case, a maximum lifetime of 90 days is applied, following the definition used by the open-source cloud security tool Prowler. Instances older than this threshold may not be regularly updated and could rely on deprecated dependen-

cies, increasing the likelihood of associated CVEs. This creates the threat that attackers exploit outdated software components, potentially affecting the confidentiality, integrity, and availability of data and services. As all three dimensions can be affected, the potential impact level is assessed as high. Together with the medium exposure level, this results in an overall high severity level.

- *Instances with outdated templates:* Another possible threat is the use of outdated instance templates. Templates serve as the basis for creating new instances, defining their configuration and software stack. If templates are not kept up to date, newly created instances may inherit outdated operating systems or dependencies, which can increase the likelihood of known CVEs being present from the outset. This threat can affect the confidentiality, integrity, and availability of data and services, as attackers may target vulnerabilities introduced by outdated templates. Since all three security dimensions are potentially impacted, the potential impact level is considered high. Together with the medium exposure level, this results in an overall high severity level.
- *Instances exposed without restriction on layer three:* A possible threat occurs when an instance is exposed to `0.0.0.0/0`. This configuration allows access from any network without restriction, which results in an expanded attack surface. Such exposure can affect the confidentiality, availability, and integrity of data and services, as adversaries might find possible exploits on services that were not meant to be exposed. The assessed potential impact level is high, as exploitation could disrupt or compromise services. The resource may be confirmed to be publicly exposed on layer three, but layer four restrictions may still apply. Therefore, the exposure level remains medium. This results in an overall high score.
- *Instances exposed without restriction on layer three and four:* Another threat exists when an instance is made publicly available on all ports and to all IP ranges. While this does not necessarily mean the instance is accessible from every network, it does increase the range of reachable services. This raises the potential for adversaries to discover and exploit running services, again potentially impacting the confidentiality, availability, and integrity of the system. The potential impact level is considered high, as multiple services may be compromised. The resource may be confirmed to be publicly exposed on layers three and four without restrictions. Therefore, the exposure level is increased to high. This results in an overall critical score.
- *Instances expose specific service ports to the internet:* Another possible threat occurs when

instances expose specific ports to all IP ranges. The specified ports are typically used for remote access protocols, file-transfer protocols, or database services, which can directly affect the confidentiality, integrity, and availability of data and services when exposed. The ports themselves are defined in the threats for the category *Security Groups*, but their evaluation is also of relevance in this category. The reason for this evaluation is that the security groups are confirmed to be assigned to a public-facing resource, meaning a higher overall score is warranted. Public exposure of such ports increases the attack surface, as adversaries may attempt to exploit known vulnerabilities associated with these services. This may affect the confidentiality, availability, and integrity of data and services. Therefore, the potential impact level is assessed as high. With unrestricted exposure to the internet, an increased exposure level of high is assessed. This results in an overall critical severity level.

- *Unreachable instances:* An instance without a public IP address and without an assigned private network is unreachable. In this state, the instance cannot be accessed to deliver or support services, effectively rendering it unusable. This impacts financial resources, as the consumer continues to pay for an instance that provides no operational value. Since the threat primarily concerns financial inefficiency and does not directly affect the confidentiality, integrity, or availability of data and services, the potential impact level is assessed as low. The exposure level is also low, as the resource is confirmed to be unreachable. This results in an overall informational score.
- *Resource overprovisioning:* Large instance types are instance types that must be explicitly unlocked by Exoscale for a given account. Their use does not pose a direct security threat, but it is worth noting from an informational perspective. Running workloads on such instances may result in an over-provisioning of resources, leading to the inefficient use of resources. However, this concern is operational rather than security-related. Since no direct potential impact on the confidentiality, integrity, or availability of data and services exists, the potential impact level is assessed as low. Together with the medium exposure level, this results in an overall low severity level.

Load Balancer

Resource Configuration Options: When creating load balancers, the region and the name must be selected, and a description can be provided. The chosen name must be individual in the organizational context.

After the creation of a load balancer, services can be added. Each service must be given a name that must be unique in the load balancer context. Furthermore, the instance pool, service strategy, protocol, service port, and target port must be chosen along with an optional description of the service. The health check settings are identical to those in the category *Elastic IP*. Once a service is created, the instance pool cannot be altered, in contrast to the other settings. An instance pool cannot be deleted when it is referenced by a service.

Exposure Level: Load balancers are always public, but rely on instance pools and their member instances for traffic destinations, which can have restricted public exposure through the use of security groups. Therefore, the exposure level for this resource type is set to medium.

Possible Threats:

- *Failed health checks block traffic:* If a load balancer's health check fails, traffic is not forwarded to its underlying instances, directly impacting the availability of the associated services. As availability is confirmed to be affected, the potential impact level is high. Together with the medium exposure level, this results in an overall high severity level.
- *Health checks blocked by security groups:* If a load balancer is associated with an instance pool whose member instances are assigned a security group that blocks the health check, the health check will fail, and no traffic will be forwarded to the underlying resources. This directly impacts the availability of the services relying on those instances. Since availability is confirmed to be affected, the potential impact level is high. Together with the medium exposure level, this results in an overall high severity level.

Private Networks

Resource Configuration Options: When creating a private network, a name, a region, and a type must be supplied. Optionally, a description can be added. Should the type be managed, a start and end IPv4 address and netmask must be chosen.

Exposure Level: Private networks cannot be exposed to the public. Therefore, the exposure level is low.

Possible Threats:

- *Private network exhausted IP leases:* A potential threat in private networks arises when subnets become nearly full, especially in environments where infrastructure is automatically deployed. For this analysis, a universal tolerance level of 75% is defined for the associated controls. If no IP addresses are available for new instances, additional resources cannot be provisioned, which can prevent services from scaling as needed. Therefore, the availability of services would be impaired, warranting a high potential impact level. Together with the low exposure level, this results in an overall medium severity level.

Security Groups

Resource Configuration Options: When creating a security group, a name must be specified, and a description can be specified. After the creation of a group, predefined or custom rules can be added. Exoscale offers a predefined rule to allow Internet Control Message Protocol (ICMP), SSH, or Remote Desktop Protocol (RDP). When creating a customized rule, the traffic flow direction, protocol, and source type must be specified. Supported protocols are Transmission Control Protocol (TCP), User Datagram Protocol (UDP), ICMP, Internet Control Message Protocol for the Internet Protocol Version 6 (ICMPv6), Authentication Header (AH), Encapsulating Security Payload (ESP), Generic Routing Encapsulation (GRE), and IP-in-IP (IPIP). The supported source types are Classless Inter-Domain Routing (CIDR), other security groups, and public security groups. Public security groups allow ports from load balancer healthchecks, object storage endpoints, and API servers from SKS. Depending on the protocol and source type, a source in CIDR-format, and a port range must be provided.

Exposure Level: Security groups are used to limit access to other resources, which may be public. However, they are not publicly exposed themselves which results in a low exposure level.

Possible Threats:

- *Security groups allow unrestricted ingress traffic on layer three:* A possible threat occurs when a security group contains rules that allow access from `0.0.0.0/0`. While this configuration itself does not guarantee exposure, since a security group may exist without being associated with an active resource, it creates the potential for instances to be publicly reachable without restrictions once the group is applied. This expands the possible attack surface

and could affect the confidentiality, availability, and integrity of data and services if adversaries are able to exploit services unintentionally exposed. The assessed potential impact level is high, as exploitation could compromise or disrupt potentially associated services. Together with the low exposure level, this results in an overall medium severity level.

- *Security groups allow ingress traffic from class B network:* A possible threat also arises when a security group permits ingress from networks with a prefix length of up to 16. A network of this size contains 65,536 IPv4 addresses. This threshold is chosen because a prefix length of 16 corresponds to the traditional class B network size and still admits tens of thousands of source addresses, warranting separate attention from smaller networks. Allowing access from this range, therefore, creates the potential for external reachability once the group is applied, increasing the likelihood that exposed services could be probed or exploited, with possible effects on confidentiality, integrity, and availability. The potential impact level is assessed as high, as successful exploitation could compromise associated services. Together with the low exposure level, this results in an overall medium severity level.
- *Security groups allow unrestricted ingress traffic on layer four:* A possible threat occurs when a security group contains rules that allow ingress traffic on all ports. This configuration does not automatically imply exposure, since the security group may exist without being associated with an active resource, but once applied, it removes any layer four restriction of the services that can be reached externally. Allowing access from this range, therefore, creates the potential for external reachability once the group is applied, increasing the likelihood that exposed services could be probed or exploited, with possible effects on confidentiality, integrity, and availability. The potential impact level is therefore assessed as high. Together with the low exposure level, this results in an overall medium severity level.
- *Security groups allow ingress traffic to specific ports:* A further threat arises when security groups allow inbound access to specific ports that are commonly associated with databases, authentication protocols, management interfaces, or remote access services. An overview is provided in Table 5.1, which contains the ports, their protocols, category, and a short threat description. Exposing such ports directly to the internet can significantly increase the attack surface, as these services may rely on default configurations, no authentication, or unencrypted communication channels. If exploited, adversaries could gain unauthorized access, manipulate data, or disrupt services, which would impact the confidentiality, integrity,

and availability of affected systems. The assessed potential impact level is high due to the potential severity of compromise. Together with the low exposure level, this results in an overall medium severity level.

- *Unused security groups:* Unused security groups may remain in an account without being linked to any active resource. In such cases, they neither introduce a direct threat to confidentiality, integrity, or availability, nor do they generate additional costs. Their existence may be the result of infrastructure modifications, such as the removal of instances without corresponding cleanup, or from temporary configurations created during testing and development. While unused groups are not inherently dangerous, they can reduce transparency and complicate the management of network policies, which may affect the clarity of the overall security posture. As no potential impact on services or data exists, the assessed potential impact level is low. Together with the low exposure level, this results in an overall informational severity level.

Port	Protocol	Category	Threat Description
27017, 27018	MongoDB	Database	MongoDB should not be exposed without proper authentication. Threats include data theft, corruption, or unauthorized queries.
7199, 9160, 8888	Cassandra	Database	Public access to Cassandra can lead to manipulation or leakage of large data sets due to missing authentication mechanisms and can lead to data exposure.
9092	Kafka	Database	Exposing Kafka brokers allows adversaries to intercept or inject messages, potentially corrupting event-driven systems.
11211	Memcached	Database	Public Memcached is vulnerable to amplification attacks and data leakage from cached memory.
3306	MySQL	Database	Public-facing MySQL exposes sensitive data and is a target for brute force attacks.

Continued on next page

Continued from previous page

Port	Protocol	Category	Threat Description
1521, 2483	Oracle	Database	Oracle DB ports are potential targets for exploitation.
5432	PostgreSQL	Database	PostgreSQL exposure can lead to unauthorized data access, corruption, or service outage.
6379	Redis	Database	Redis does not enforce authentication by default, hence, public exposure can lead to data tampering or remote code execution.
1433, 1434	SQL Server	Database	Exposed Microsoft SQL ports can be subject to credential attacks and can lead to the compromise of backend systems.
88, 464, 749, 750	Kerberos	Kerberos/ LDAP	Public Kerberos and Lightweight Directory Access Protocol (LDAP) ports can lead to the theft of login details, replay attacks, and unauthorised authentication.
389, 636	LDAP	Kerberos/ LDAP	Exposing LDAP enables unauthorized directory queries or modifications, potentially impacting integrity and confidentiality.
9200, 9300, 5601	Elasticsearch / Kibana	Elasticsearch/ Kibana	Public Elasticsearch and Kibana ports may allow unauthorized data queries or manipulation.
22, 23, 512, 514, 992, 3389	Remote Access Protocols	Management	Remote access protocols are potential entry points for attackers. Threats include credential brute forcing and system takeover.

Continued on next page

Continued from previous page

Port	Protocol	Category	Threat Description
20, 21	FTP	File Transfer	Exposing FTP can allow unencrypted credential transmission and data transfer interception.
25, 110, 143	SMTP, POP3, IMAP	Email Services	Public exposure of mail protocols can enable unauthorized relay, credential harvesting, or interception of communication.

Table 5.1: Ports and associated threats when publicly accessible

SKS

Resource Configuration Options: Upon creation of an SKS cluster, the region where the cluster resides, the cluster name, service level, control plane version, and CNI must be specified. By default, the service level is set to use the starter-plan, the most recent control plane version is selected, and Calico is used as the cluster CNI. A security group is automatically created based on the chosen CNI, and the cluster additions, exoscale-cloud-controller, exoscale-container-storage-interface, and metrics-server, are enabled. Optionally, a description can be provided, and automatic upgrades to the latest control plane patch version can be enabled.
After a cluster is created, a node pool can be added. When creating a node pool, the same settings as in the subcategory *Instance Pools* can be configured, except for the region, which must match the cluster's region and is therefore already set. It is also possible to rotate the Cloud Controller Manager (CCM) and Container Storage Interface (CSI) credentials, as well as the operators' Certificate Authority (CA). Additionally, the Kubeconfig and cluster CA can be retrieved, and the cluster service level can be upgraded.

Exposure Level: If consumers wish to expose a service publicly, a load balancer would have to be used. Since the control plane is fully managed by Exoscale and its status cannot be verified by consumers, the exposure level for this resource type is considered low.

Possible Threats: Instance pools and nodes within SKS clusters are already evaluated by the controls in the respective subcategories *Instance Pools* and *Instances*, as they are fully integrated

into the Exoscale environment. Therefore, threats related to underlying compute resources, such as outdated operating systems, exposed ports, or insecure configurations, are covered by the corresponding instance and instance pool controls.

- *Automatic updates for SKS:* A potential threat arises if automatic updates for the SKS control plane are disabled. In this case, the cluster may fail to receive the latest Kubernetes patch versions, leaving it exposed to known vulnerabilities and security issues. The assessed potential impact level is high, as exploitation could compromise or disrupt the control plane and potentially affect workloads. Together with the low exposure level, this results in an overall medium severity level.
- *Outdated SKS control plane:* Another possible threat occurs when the used SKS control plane version is outdated. In this analysis, the term outdated refers to versions that are no longer offered by Exoscale for newly created clusters. Continuing to operate on such a version means missing critical security patches and stability improvements, exposing workloads to vulnerabilities and operational issues. The assessed potential impact level is high because it could directly affect the confidentiality, availability, and integrity of workloads. Together with the low exposure level, this results in an overall medium severity level.

SSH Keys

Resource Configuration Options: When creating a new key, a key name and the content of the public key must be specified.

Exposure Level: SSH keys are not a public resource in Exoscale. After their creation, the content of the public key cannot be viewed. Therefore, the exposure level of this resource is assessed as low. However, it must be considered that keys might be exposed through unintentional leaks on version control platforms such as GitHub.[2]

Possible Threats:

- *SSH key rotation:* A possible threat arises when SSH keys are used for a period of time. For this analysis, a threshold of 180 days was defined. In cloud environments, keys can be accidentally exposed, an example being the exposure through public repositories. The likelihood of compromise increases the longer they remain in use without rotation. The

[2] https://github.com/

assessed potential impact level is high, as a compromised key would allow direct access to associated instances, affecting confidentiality and integrity. Together with the low exposure level, this results in an overall medium severity level.

- *Unused SSH keys:* A possible threat arises when SSH keys are unused. Such keys may remain from past administrative activities or former employees, and while they do not pose an immediate threat if not associated with active instances, they can introduce the threat of accidental reuse. If reassigned, they could enable unauthorized access. This could impact the confidentiality or integrity of affected resources. Therefore, the assessed potential impact level is high. Together with the low exposure level, this results in an overall medium severity level.

Templates

Resource Configuration Options: When creating a custom template, the region, name, publicly accessible Uniform Resource Locator (URL) of the template file, corresponding checksum, login username, and boot mode must be specified. Optionally, a description can be added, and SSH as well as password-based authentication can be enabled.

Exposure Level: When creating templates as a consumer, the use of these templates is restricted to the consumer's organization. Therefore, it is not public, which results in a low exposure level.

Possible Threats: No specific threats are cited for custom templates in this analysis in relation to the underlying cloud services. The security and stability of custom templates depend entirely on the content and ongoing maintenance of the uploaded image, which is not part of the base cloud services and, therefore, out of scope.

5.1.2 Storage

This subsection analyzes the configuration options of the *Storage* category, outlines their resource-specific exposure levels, and presents the identified threats together with their assigned severity ratings.

Object Storage

Resource Configuration Options: When creating a bucket, the region and name must be specified. Optionally, the object lock capability can be enabled. After creation, consumers can configure replication, set a retention period for object lock, enable the Content Delivery Network (CDN) functionality, define CORS settings, and manage ACL rules. For access control, either preconfigured (canned) ACLs or custom ACLs can be applied. The available canned ACLs are *private*, *public-read*, *public-read-write*, and *authenticated-read*. Custom ACLs can be defined for the organization owner, authenticated users, or all users, with possible permissions including *read*, *write*, *read Access Control Policy (ACP)*, *write ACP*, and *full control*.

Exposure Level: All objects stored on the Exoscale platform can be exposed publicly. However, access can be restricted through ACLs or CORS rules, which results in a medium exposure level.

Possible Threats:

- *Bucket allows full control to all users:* A possible threat occurs when a bucket is configured with an ACL that grants full control to all users. This permission allows any party to read, write, and modify access policies, which may result in unauthorized disclosure, data manipulation, or deletion of objects. The assessed potential impact level is high, as adversaries would be able to compromise both the confidentiality, availability, and integrity of stored data. Since the exposure is confirmed to apply to all users without restriction, the exposure level is also high. This results in an overall score of critical.
- *Bucket allows read permissions to all users:* Granting read access to all users allows anyone on the internet to retrieve objects stored in the bucket without authentication. This configuration can lead to unintended data exposure, loss of confidentiality, and potential regulatory non-compliance. The assessed potential impact level is high, as sensitive information may be leaked. Since the exposure is confirmed to apply to all users without restriction, the exposure level is also high. This results in an overall score of critical.
- *Bucket allows read and write permissions to all users:* Buckets with read-write permissions for all users are exposed to both unauthorized data access and data modification. This configuration enables adversaries not only to read but also to add, alter, or delete objects. Such actions could disrupt business operations, cause data loss, or introduce malicious files into applications. The assessed potential impact is therefore high. Since the exposure

is confirmed to apply to all users without restriction, the exposure level is also high. This results in an overall score of critical.

- *Bucket allows unrestricted CORS access:* A threat arises when buckets are configured with unrestricted CORS rules using wildcards. This setting allows any domain to issue cross-origin requests, which may enable malicious websites to interact with bucket data if credentials are exposed in client environments. The assessed potential impact level is high, as exploitation directly threatens the confidentiality of stored data. Since the exposure is confirmed to apply to any domains, the exposure level is also high. This results in an overall score of critical.
- *Disabled versioning on buckets:* If bucket versioning is disabled, overwritten or deleted objects cannot be restored. This increases the threat of data loss or deliberate data deletion without recovery options. While this does not increase the attack surface, it directly affects data availability and resilience. The assessed potential impact is high, as the loss of data availability could disrupt operations. Together with the medium exposure level, this results in an overall high severity level.

5.1.3 DBaaS

This subsection analyzes the configuration options of the *DBaaS* category and outlines their resource-specific exposure levels. It presents the identified threats together with their assigned severity ratings, focusing on the SQL databases PostgreSQL and MySQL.

MySQL

Resource Configuration Options: When creating a MySQL service, the consumer must specify the region in which the service is deployed, the resource name, and the chosen service plan. Optionally, IP filters can be defined to restrict incoming IPv4 connections. It is also possible to migrate data from an existing external database. Custom settings specific to MySQL can only be configured through the API or CLI. After creation, termination protection can be enabled to prevent accidental deletion of the service. Additional configuration options include managing IP filters, retrieving logs, monitoring metrics, and scheduling maintenance windows. External integrations with services such as Datadog, Elasticsearch, OpenSearch, Prometheus, and syslog are supported.

Consumers can also add read replicas, create new users with corresponding authentication methods, and adjust database parameters. Once defined, database parameters cannot be removed, and certain modifications may trigger a service restart.

Exposure Level: MySQL services can be configured to be publicly exposed, while also allowing the filtering of incoming IPv4 connections. Therefore, the exposure level is medium.

Possible Threats:

- *Disabled termination protection:* If termination protection is not enabled, database services may be unintentionally terminated. While an accidental termination does not compromise confidentiality or integrity, it directly impacts availability by potentially interrupting critical services and workflows, and restoration may rely on backups and recovery procedures. The potential impact level is therefore assessed as high. Together with the medium exposure level, this results in an overall high severity level.
- *Unrestricted IP filter:* If the IP filter allows access from all addresses, the database service is exposed to the entire Internet. This increases the threat of unauthorized access, potentially affecting the confidentiality, integrity, and availability of stored data. Therefore, the potential impact level is assessed as high. As the resource is confirmed to be public with no restrictions in this case, the exposure level is increased to high, which results in an overall critical severity level.
- *Enforced SSL encryption:* If SSL encryption is not enforced, data transmitted between clients and the database may be transmitted unencrypted. This could allow interception or modification of data in transit, directly affecting confidentiality and integrity. The potential impact level is therefore assessed as high. Together with the medium exposure level, this results in an overall high severity level.
- *Resource unreachable:* If a database instance becomes unreachable due to missing IP filter configurations, dependent workflows may be interrupted. While confidentiality and integrity remain unaffected, availability is directly impacted. Additionally, resources remain allocated without delivering operational benefit, leading to the use of financial resources. The potential impact level is therefore assessed as high. Together with the medium exposure level, this results in an overall high severity level.

PostgreSQL

Resource Configuration Options: When creating a PostgreSQL service, the user must select the region, resource name, and service plan. Optionally, IP filters can be configured to restrict incoming IPv4 connections. Migration from an existing external database is also supported. Database-specific settings can only be modified via the API or CLI. After creation, termination protection can be enabled to prevent accidental deletion. Consumers may also configure IP filters, retrieve logs, monitor metrics, schedule maintenance windows, and add external integrations such as Datadog, Elasticsearch, OpenSearch, Prometheus, or syslog. Additional options include the creation of connection pools, adding read replicas, managing users, and updating database parameters. Certain changes may trigger a service restart, and once set, parameters cannot be removed.

Exposure Level: PostgreSQL services can be configured to be publicly exposed, while also allowing the filtering of incoming IPv4 connections. Therefore, the exposure level is medium.

Possible Threats:

- *Disabled termination protection:* If termination protection is not enabled, PostgreSQL services may be unintentionally deleted. While such accidental termination does not compromise confidentiality or integrity, it directly impacts availability by potentially interrupting critical services and workflows, and restoration may rely on backups and recovery procedures. The potential impact level is therefore assessed as high. Together with the medium exposure level, this results in an overall high severity level.
- *Unrestricted IP filter:* If the IP filter allows access from all addresses, the PostgreSQL service is exposed to the internet. This increases the threat of unauthorized access, potentially affecting confidentiality, integrity, and availability. Therefore, the potential impact level is assessed as high. As the resource is confirmed to be public with no restrictions in this case, the exposure level is increased to high, which results in an overall critical severity level.
- *Enforced SSL encryption:* If SSL encryption is not enforced, data transmitted between clients and the database may be sent unencrypted. This could allow interception or modification of data in transit, directly affecting confidentiality and integrity. The potential impact level is therefore assessed as high. Together with the medium exposure level, this results in an overall high severity level.
- *Resource unreachable:* If a PostgreSQL instance becomes unreachable due to missing IP

filter configurations, dependent workflows may be interrupted. While confidentiality and integrity remain unaffected, availability is directly impacted. Additionally, resources remain allocated without delivering operational benefit, which can result in the use of financial resources. The potential impact level is therefore assessed as high. Together with the medium exposure level, this results in an overall high severity level.

- *Deprecated password hash:* If the deprecated `password_encryption` option is set to use the `MD5` hash for PostgreSQL users, an attacker who obtains the password hash from the server may compromise the stored credentials [47]. This could lead to unauthorized access, affecting confidentiality, integrity, and availability. The potential impact level is therefore high. Together with the medium exposure level, this results in an overall high severity level.

5.1.4 IAM

This subsection analyzes the configuration options of the *IAM* category, outlines their resource-specific exposure levels, and presents the identified threats together with their assigned severity ratings.

Users and Keys

Resource Configuration Options: When adding a user, a role, and a mail address must be specified. An invite mail is then sent to the specified email. The user can set his password, enable MFA, and modify his account information. MFA can be mandated, so that all users within an organization must have a second factor registered.

Exposure Level: User accounts and API keys are not directly exposed by Exoscale, but they are required to access the publicly available administration portal or API to perform their permitted actions. Beyond this, they are also exposed to individuals handling them in the course of their tasks, making them susceptible to phishing attacks or unintentional disclosure, for example, through version control platforms. For these reasons, the exposure level is assessed as high.

Possible Threats:

- *Enforcing MFA:* Suganya Subramani et al. recognize MFA as a vital security technique that enhances the implementation of access controls [48]. The BSI recommends the use of

two-factor authentication for administrative platforms. Therefore, the failure to use MFA constitutes a violation of existing best practices. The potential compromise of user accounts or API keys could result in unauthorized access to cloud resources, threatening data confidentiality, integrity, and availability, warranting a high potential impact level. Together with the high exposure level, this results in an overall critical severity level.

Roles and Policies

Resource Configuration Options: When creating a role, a name and description must be defined. Subsequently, it can be specified whether the role is editable, which is an attribute that cannot be modified later. At this stage, role permissions such as `bypass-governance-retention` and `reset-iam-organization-policy` can also be enabled, and the associated policy must be defined. Policies may be created in the user interface mode or in the advanced mode, which exposes the underlying JavaScript Object Notation (JSON) definition. The permission `reset-iam-organization` grants the role holder the ability to reset the organization policy, to which all requests are subject. The permission `bypass-governance-retention` allows actions that would otherwise be restricted by retention rules, for example, in object storage. If no service-specific rules are defined, the default service strategy applies to all services. When specified, service-specific strategies override the default strategy. In cases where no explicit rules are defined for a service, all actions in the respective category remain unrestricted.

Exposure Level: For IAM roles to take effect, they must be associated with a user account or API key. However, they are not necessarily attached to accounts or keys and are not publicly exposed by Exoscale. Therefore, the exposure level of roles and policies is assessed as low.

Possible Threats:

- *Role allows full control over IAM:* If a role is configured with permissions that allow full control over the IAM service, this grants the role holder unrestricted ability to modify or delete organizational policies, roles, and user assignments. Such access directly threatens the integrity of the entire access control system, as it allows privilege escalation, circumvention of governance restrictions, and the removal of safeguards such as mandatory MFA. The potential impact level is high, as a compromise would affect the confidentiality, integrity, and

availability of all organizational resources. Together with the low exposure level, this results in an overall medium severity level.

- *Role allows full control over account:* Another configuration would be the granting of universal access to all services and actions within the organization. This undermines the principle of least privilege and warrants a high potential impact level, as a compromise would affect the confidentiality, integrity, and availability of all organizational resources. Together with the low exposure level, this results in an overall medium severity level.

5.2 Exoscan: Prototype for Security Analysis of Exoscale Cloud

The prototype receives the name Exoscan, highlighting its primary purpose of scanning for configuration weaknesses within the Exoscale cloud platform. Its design objectives focus on ensuring logging capabilities, employing secure authentication mechanisms, and a modular architecture. Furthermore, the prototype is conceived with extensibility in mind, allowing future enhancements and the integration of additional functionalities. The objective of the first version is to automatically detect insecure configurations and constellations and to provide the user with a report containing the triggered controls. The previously identified threats are translated into controls, which form the basis of the scanning activities.

5.2.1 Development Environment and Dependencies

The implementation was carried out using Python 3.13.2. To support interaction with Exoscale services and facilitate configuration, logging, and validation tasks, several external libraries were utilized. Table 5.2 provides an overview of the utilized software dependencies and their corresponding versions.

Package	Version
requests-exoscale-auth	$\geq$ 1.1.2
boto3	$\geq$ 1.40.16
importlib	$\geq$ 1.0.4
pydantic	$\geq$ 2.0
python-dotenv	$\geq$ 1.0
requests	$\geq$ 2.32.3

Table 5.2: Python dependencies used in the implementation

5.2.2 Exoscan's Architecture

Listing 1 illustrates the directory layout of the prototype. The root directory contains configurations for logging, argument parsing, and the authentication mechanism, as well as the exoscan subdirectory, which holds all files required for the implementation of controls. Within this subdirectory, functions are defined to dynamically retrieve controls from their respective subcategories and execute the corresponding scanning logic. Furthermore, the subdirectory is again structured into directories representing the analyzed services. These directories reflect the individual subcategories of each service and contain the respective control directories. Each control directory comprises a Python file implementing the scanning logic and a JSON file providing the associated metadata.

```
exoscan/
|-- exoscan/
|   |-- __main__.py
|   |-- lib/
|   |   |-- banner.py
|   |   |-- controls/
|   |   |   |-- controls_loader.py
|   |   |   |-- execute_controls.py
|   |   |   |-- models.py
|   |   |   |-- utils.py
|   |   |   |-- compute/
|   |   |   |   |-- block_storage/
|   |   |   |   |   |-- compute_block_storage_no_snapshots
|   |   |   |   |   |   |-- compute_block_storage_no_snapshots.py
|   |   |   |   |   |   |-- compute_block_storage_no_snapshots.metadata.json
```

```
|   |   |   |   |   '-- ...
|   |   |   |   |-- elastic_ip/
|   |   |   |   |-- instance_pools/
|   |   |   |   |-- instances/
|   |   |   |   |-- load_balancer/
|   |   |   |   |-- private_networks/
|   |   |   |   |-- security_groups/
|   |   |   |   |-- sks/
|   |   |   |   |-- ssh_keys/
|   |   |   |   '-- templates/
|   |   |   |-- dbaas/
|   |   |   |   |-- general/
|   |   |   |   |-- mysql/
|   |   |   |   '-- postgresql/
|   |   |   |-- iam/
|   |   |   |   |-- roles/
|   |   |   |   '-- users/
|   |   |   '-- storage/
|   |   |   |   '-- buckets/
|-- parser/
|   '-- parser.py
|-- log_conf/
|   '-- logger.py
|-- provider/
|   '-- exoscale_provider.py
|-- exoscan.py
|-- exoscan.log
|-- README.md
'-- requirements.txt
```

Listing 1: Directory structure of Exoscan

5.2.3 Logging Capabilities and Authentication Procedure

As can be seen in Listing 2, based on the arguments provided at runtime, the prototype logs events of the levels *Warning*, *Error*, and *Critical* to the CLI. In addition, file-based logging is implemented, which consistently records all log levels: *Debug*, *Info*, *Warning*, *Error*, and *Critical*. The function `set_logging_config` is executed in the main module at the start of the prototype, ensuring centralized configuration. Other modules requiring logging import the logger from this file, thereby maintaining a unified and consistent logging structure across the implementation.

```
def set_logging_config(verbose):
    formatter = logging.Formatter(
        "{asctime} - {levelname} - {message}",
        style="{",
        datefmt="%Y-%m-%d %H:%M",
    )

    logging_handlers = []

    console_handler = logging.StreamHandler()
    console_handler.setFormatter(formatter)
    if verbose: console_handler.setLevel(10)
    else: console_handler.setLevel(30)

    file_handler = logging.FileHandler("exoscan.log", mode="a", encoding="utf-8")
    file_handler.setFormatter(formatter)
    file_handler.setLevel(10)

    logging_handlers.append(console_handler)
    logging_handlers.append(file_handler)

    logging.basicConfig(handlers=logging_handlers, level=logging.INFO)


logger = logging.getLogger()
```

Listing 2: Logging configuration

To interact with the API of Exoscale, a form of authentication is required. The package requests-exoscale-auth provides functionality to handle this interaction securely. The API key and secret are supplied via environment variables, ensuring that sensitive credentials never leave the local system. Although the use of a credentials file was considered, this approach was discarded to mitigate the risk of accidental uploads to version control platforms such as GitHub. The corresponding code can be inspected in Listing 3.

```
EXOSCALE_API_KEY = os.getenv('EXOSCALE_API_KEY')
EXOSCALE_API_SECRET = os.getenv('EXOSCALE_API_SECRET')

def authenticate():
    try:
        if EXOSCALE_API_KEY == None and EXOSCALE_API_SECRET == None:
            raise ConnectionAbortedError
```

```
        else:
            auth = ExoscaleV2Auth(EXOSCALE_API_KEY, EXOSCALE_API_SECRET)
            return auth
    except (ConnectionAbortedError):
        logger.error("Authentication Methods could not find credentials in the form of env
        ↪ vars.")
        sys.exit("Authentication impossible, exiting...")
```

Listing 3: Authentication provider

5.2.4 Dynamic Discovery and Execution of Controls

The scanning process of Exoscan relies on the functions `import_all_controls`, `fetch_controls`, and `execute_controls`. Together, they ensure that all implemented controls are dynamically discovered, prepared for execution, and processed in a uniform manner.

The function `import_all_controls`, as seen in Listing 4, traverses the directory hierarchy under `exoscan.lib.controls` and identifies valid controls by evaluating their naming depth and path structure. For each control, the function collects both the control name and its path, returning them as tuples. This approach avoids hardcoding and enables the inclusion of newly added controls across services and subcategories.

```
import sys, importlib
from pkgutil import walk_packages
from log_conf.logger import logger

def import_all_controls(services: str = None) -> list[tuple]:
    try:
        controls = []
        modules = []

        module_path = f"exoscan.lib.controls"
        modules = walk_packages(
        importlib.import_module(module_path).__path__,
        importlib.import_module(module_path).__name__ + "."
        )

        for module_name in modules:
            control_module_name = module_name.name

            if (control_module_name.count(".") == 6):
```

```
20
21                  control_path = module_name.module_finder.path
22                  control_name = control_module_name.split(".")[-1]
23                  control_info = (control_name, control_path)
24                  controls.append(control_info)
25          return controls
26      except Exception as e:
27          logger.critical(f"{e.__class__.__name__}[{e.__traceback__.tb_lineno}]: {e}")
28          sys.exit("Critical error in control import, exiting...")
```

Listing 4: Function `import_all_controls`

The function `fetch_controls`, as seen in Listing 5, builds on this result by normalizing the retrieved paths into importable Python modules. To account for different operating systems, both Windows and Linux path conventions are processed and transformed into a consistent format. However, the first iteration of the prototype will only support Windows-based systems. All discovered modules are then prepared for execution, while logging provides traceability.

```
1  from exoscan.lib.controls.utils import import_all_controls
2  from log_conf.logger import logger
3
4  def fetch_controls() -> set:
5      try:
6          logger.info("Fetching controls...")
7          controls_modules_path = []
8          controls = import_all_controls()
9
10         for control in controls:
11             if "\\" in control[1]:
12                 controls_modules_path.append(
13                 '.'.join((f"{control[1]}\\{control[0]}").split("\\")[-7:]))
14             elif "/" in control[1]:
15                 controls_modules_path.append(
16                 '.'.join((f"{control[1]}/{control[0]}").split("/")[-7:]))
17         return controls_modules_path
18     except Exception as error:
19         logger.error(
20             f"{error.__class__.__name__}[{error.__traceback__.tb_lineno}] -- {error}"
21         )
22         return controls_modules_path
```

Listing 5: Function `fetch_controls`

The actual execution is performed by `execute_controls`, as seen in Listing 6. Each control module is imported dynamically, and the corresponding metadata file is constructed. The function then invokes the `execute_logic` method of the control, which applies the defined scanning procedure based on the metadata. The findings are collected and returned as a consolidated result set.

```python
import importlib
from log_conf.logger import logger
from pathlib import Path

def execute_controls(controls_modules_path):
    logger.info("Executing controls...")
    findings = []
    try:
        for module in controls_modules_path:
            control = importlib.import_module(f"{module}")

            parts = module.split(".")

            *base_path, filename = parts

            metadata_path = Path("/".join(base_path)) / f"{filename}.metadata.json"
            control_findings = control.execute_logic(metadata_path)
            if control_findings:
                findings.extend(control_findings)
        return findings
    except Exception as error:
        logger.error(
            f"{error.__class__.__name__}[{error.__traceback__.tb_lineno}] -- {error}"
        )
```

Listing 6: Function `execute_controls`

In the prototype, these functions are called sequentially in the main module: `fetch_controls` discovers and prepares the controls, `execute_controls` performs the scanning, and the resulting findings are aggregated. This workflow ensures that newly implemented controls are automatically integrated into the scanning process, reflecting the design objectives of modularity and extensibility.

5.2.5 Design of Controls and Findings

In each control directory, a JSON file containing the control's metadata resides alongside a Python file implementing its logic. The metadata file contains a description, a summary of the underlying threat, a recommendation, the severity, and a unique control name. The control name follows a structured naming convention, starting with the category and the subcategory, followed by a descriptive identifier. A representative metadata file is shown in Listing 7.

```
{
    "control_name": "compute_block_storage_unused_volume",
    "control_description": "Check if any block storage volume is unused.",
    "threat": "Having unused block storage volumes increases costs without providing actual
    ↪  functionality.",
    "recommendation": "Review if unused block storage volumes are still relevant.",
    "severity": "medium"
}
```

Listing 7: Example of a control metadata file

The file containing the programming logic consists of the imports that are necessary for the execution and the function containing the logic itself. Each control exposes a single function, `execute_logic`, which is invoked by the overarching `execute_controls` function. The overall structure of each control adheres to a consistent pattern, starting with an informational log entry, followed by a call to the corresponding inventory function to retrieve the resources subject to evaluation. If no resources are found, the control is skipped. Otherwise, the defined logic is applied to each resource, and whenever a resource fails to meet the specified conditions, a finding is generated. These findings are then returned to the calling function and incorporated into the aggregated results.

```
import sys
from exoscan.lib.controls.compute.block_storage.inventory import get_block_storage_volumes
from exoscan.lib.controls.models import Finding
from log_conf.logger import logger

def execute_logic(metadata_path):
    logger.info("Executing Control: compute_block_storage_unused_volume")
    try:
        all_volumes = get_block_storage_volumes()
        if not all_volumes:
```

```
11              logger.info("No Volumes found. Skipping Control
                ↪  compute_block_storage_unused_volume...")
12              return
13          findings = []
14          unused_volumes = []
15          for volume in all_volumes.block_storage_volumes:
16              if not volume.instance.id and volume.block_storage_snapshots:
17                  unused_volumes.append(volume.name)
18
19          if unused_volumes:
20              volumes_str = "\n - ".join(set(unused_volumes))
21              findings.append(Finding.from_metadata(
22                          metadata_file= metadata_path,
23                          resource_description=f"Volumes:\n - {volumes_str}"
24                      ))
25          return findings
26
27      except Exception as error:
28          logger.error(f"{error.__class__.__name__}[{error.__traceback__.tb_lineno}] --
            ↪  {error}")
29          sys.exit(1)
30      return
```

Listing 8: Logic of control `compute_block_storage_unused_volume`

In order to represent and manage the results of executed controls, Exoscan employs a set of data models that standardize the structure of findings. These models ensure that each finding is consistently enriched with metadata, severity, and a clear description of the affected resource. As can be seen in Listing 9, the severity levels are defined in the class `Severity`, which specifies five categories, providing a uniform classification scheme for identified issues.

```
1  class Severity(str, Enum):
2      critical = "critical"
3      high = "high"
4      medium = "medium"
5      low = "low"
6      informational = "informational"
7
```

Listing 9: Class `Severity`

The `ControlMetadata` class encapsulates essential information about each control, including its name, description, associated risk, recommendation, and severity. This metadata is typically loaded from the corresponding API file via the `from_file` method, ensuring coupling between the logic of a control and its descriptive context. The definition of the class is depicted in Listing 10.

```python
class ControlMetadata(BaseModel):
    severity: Severity
    control_name: str
    control_description: str
    risk: str
    recommendation: str

    @classmethod
    def from_file(cls, path: str) -> "ControlMetadata":
        with open(path) as f:
            raw = json.load(f)
        return cls.model_validate(raw)
```

Listing 10: Class `ControlMetadata`

Findings themselves are represented by the `Finding` class, which combines the control metadata with a description of the specific resource that triggered the finding set in the respective control's logic. The class definition can be seen in Listing 11. The class has a `from_metadata` constructor to create a new finding from the metadata file and a resource description. For reporting purposes, the `format_finding` method provides a structured textual representation of the finding, while `print_finding` outputs it directly, ensuring consistency in presentation across the prototype. Through this layered design, findings are standardized and integrated with their metadata definitions, enabling a systematic reporting of found threats.

```python
class Finding(BaseModel):
    ControlMetadata: ControlMetadata
    resource_description: str #refactor so it is ResourceDescription with an alias

    @classmethod
    def from_metadata(cls, metadata_file: str, resource_description: str):
        metadata_path = Path(metadata_file)
        with open(metadata_path, "r") as f:
            meta_data = json.load(f)

        control_metadata = ControlMetadata.model_validate(meta_data)
```

```
12          return cls(ControlMetadata=control_metadata,
            ↪  resource_description=resource_description)
13
14      def format_finding(self) -> str:
15          meta = self.ControlMetadata
16          lines = [
17              "-" * 80,
18              f"{meta.severity.upper()} - {meta.control_name} - {self.resource_description}",
19              "",
20              f"  - Description: {meta.control_description}",
21              f"  - Risk: {meta.risk}",
22              f"  - Recommendation: {meta.recommendation}",
23              ""
24          ]
25          return "\n".join(lines)
26
27      def print_finding(self) -> None:
28          print(f"{self.format_finding()}")
29
```

Listing 11: Class `Finding`

5.2.6 Inventory Generation and Resource Representation

When a control requires resource information, it invokes the inventory function for the respective resource type. Each subcategory within the defined services provides such an inventory function, which manages the interaction with the API to list available resources and, where necessary, retrieve additional details. An example inventory function is shown in Listing 12. The module `tempfile` is used to create temporary cache files that exist only for the lifetime of the Python process. The responses retrieved from the API are stored in this cache, ensuring that repeated requests from multiple controls do not trigger redundant API calls.

When querying the API, an initial request returns an overview of the available resources. To obtain detailed information, each resource must be queried individually by specifying its unique identifier. Additionally, when resources are local to their regions, all API queries must contain the desired region. If the inventory function is called with a specific resource identifier, a single object is returned. If no identifier is provided, the function returns a container object holding all resources of the corresponding type.

```python
import requests, os, json, sys, tempfile
from provider.exoscale_provider import authenticate, return_regions
from exoscan.lib.controls.models import PrivateNetwork, PrivateNetworkContainer
from log_conf.logger import logger

_temp_cache = tempfile.NamedTemporaryFile(
    prefix="private_networks_inventory_", suffix=".json", delete=False
)
CACHE_FILE = _temp_cache.name

def get_private_networks(
        private_network_id: str = None
) -> PrivateNetworkContainer | PrivateNetwork:
    try:
        if not os.path.exists(CACHE_FILE) or os.path.getsize(CACHE_FILE) == 0:
            logger.info("PrivateNetwork cache not found. Creating full inventory...")
            regions = return_regions()
            auth = authenticate()

            all_pn = []

            for region in regions:
                response = requests.get(
                    f"https://api-{region}.exoscale.com/v2/private-network", auth=auth
                )
                if response.status_code == 200:
                    for pn in response.json().get("private-networks", []):
                        details_response = requests.get(
                            f"https://api-{region}.exoscale.com/v2/private-network/" \
                            f"{pn['id']}",
                            auth=auth
                        )
                        if details_response.status_code == 200:
                            all_pn.append(details_response.json())

            container = PrivateNetworkContainer.model_validate({"private-networks":
            ↪  all_pn})

            with open(CACHE_FILE, "w") as f:
                json.dump(
                    container.model_dump(mode="json", by_alias=True), f, indent=2
                )

        with open(CACHE_FILE, "r") as f:
            json_data = json.load(f)
```

```
45
46          if private_network_id:
47              for pn in json_data.get("private-networks", []):
48                  if pn.get("id") == private_network_id:
49                      return PrivateNetwork.model_validate(pn)
50
51              raise Exception(
52                  f"Private Network ID '{private_network_id}' not found in cached inventory."
53              )
54
55          return PrivateNetworkContainer.model_validate(json_data)
56
57      except Exception as error:
58          logger.error(
59              f"{error.__class__.__name__}[{error.__traceback__.tb_lineno}] -- {error}"
60          )
61          sys.exit(1)
```

Listing 12: Class `Finding`

To represent resources returned by the API, classes are defined for each resource type together with a corresponding container class that holds multiple instances of the same type. The container classes inherit from a generic base class `Container`, which extends Pydantic's `BaseModel` and enables validation by attribute name as well as validation of assignments at runtime. This allows the raw API responses to be parsed directly into Python objects with their defined attributes. An example class is depicted in Listing 13, the class `PrivateNetwork` specifies the attributes of a private network resource, while `PrivateNetworkContainer` aggregates several of these objects and uses the alias `private-networks` to match the structure of the JSON response. Optional fields are modeled explicitly to account for values that may not always be present in the response. The classes and their structure follow Exoscale's API documentation [49].

```
1  class Container(BaseModel):
2      class Config:
3          validate_by_name = True
4          validate_assignment = True
5
6  class PrivateNetwork(BaseModel):
7      id: str
8      mac_address: Optional[str] = None
9
```

```
10 class PrivateNetworkContainer(Container):
11     private_networks: List['PrivateNetwork'] = Field(default=None,
       ↪  alias="private-networks")
12
```

Listing 13: Example for a resource class and its container

5.3 Limitations

Limitations arise from the design scope of the prototype, technical restrictions of the Exoscale API, and the absence of a universally applicable baseline. Each consumer may operate their environment under entirely different conditions, with varying priorities, architectures, and security requirements. As such, the tool cannot reliably define a single baseline configuration that fits all use cases, but instead evaluates against generalized conditions that may need to be adapted in practice.

First, the evaluation of security groups and IP filters cannot fully account for all cases of overly permissive rules. While obvious threats such as `0.0.0.0/0` can be detected, other large network ranges such as `/1` can still be overly permissive and pose a similar risk. Because the set of consumer-supplied networks varies, establishing a universal definition of a too permissive IP range is context dependent.

Second, unused or outdated SSH keys represent another blind spot. The Exoscale API does not expose the upload date of keys, which makes it impossible to directly assess their age. Keys that are associated with instances can only be approximated in age by considering the creation date of the respective instance, indirectly linking the key to a timeframe. However, keys that have been uploaded but are not actively attached to any instance cannot be evaluated in this way. As a result, they may persist unnoticed, even though they could pose a security risk through potential reuse.

Third, Exoscan only evaluates base-level service configurations and does not extend into the operating system level of compute instances. This means potential misconfigurations or vulnerabilities inside the virtual machines themselves are outside the prototype's scope. Similarly, not all Exoscale services are covered, as the focus was limited to a subset.

Fourth, in the case of the category *Storage*, checking ACLs for a technically unlimited amount of objects per bucket would not scale. Therefore, the prototype limits its analysis to the bucket-level configuration. More fine-grained evaluations of object-level ACLs are considered future work.

Fifth, certain service states cannot be validated due to missing API support. For example, the health check status of elastic IPs cannot be retrieved programmatically, which limits the extent to which availability-related misconfigurations can be detected.

Finally, the evaluation of cloud services in Exoscan is limited to core configuration aspects. For example, the evaluation of database services is currently limited to core configuration aspects of SQL-based databases only. Database-specific settings are not fully analyzed, even though they may contribute to the overall security posture.

In summary, the prototype does not constitute a universal solution, as the relevance of threats is always dependent on the specific context of a consumer's cloud infrastructure. These constraints, however, open up opportunities for future work, where additional service integrations and context-aware evaluations could extend the tool's capabilities.

6 Results and Discussion

This chapter presents the results of the analysis of Exoscale's services and the corresponding controls derived from the identified threats. The analysis of service configuration options, together with the implementation of the prototype, provides the foundation to answer the research questions posed in this work. The results highlight the security-relevant configuration options in the form of the defined threats and how the resulting threats can be detected automatically by means of a Python-based tool.

6.1 Identified Threats and Corresponding Controls

The formulated threats are presented in tables, split by each service category as displayed in the administration portal and defined in chapter 4. Each table includes the respective subcategory, the assigned overall severity level, and a description of the associated threat. Each control is further introduced by a row spanning all three columns, displaying the unique control identifier as implemented in the prototype. This layout ensures that every threat can be directly traced back to its control implementation while maintaining a clear overview of severity and affected service areas. These results establish a direct link between the identified threats from Exoscale's configuration options, their security implications, and the automated analysis implemented in the prototype.

6.1.1 Implemented Controls in Compute

In the *Compute* category, the analysis encompasses block storage, elastic IPs, instance pools, instances, load balancers, private networks, security groups, SKS, and SSH keys. Network exposure is examined from two perspectives. First, the configuration of security groups is assessed to determine which ports are exposed. Second, instances are evaluated in connection with their assigned security groups to reflect the operational implications of such exposure on running re-

sources. Since Exoscale services and resources are interdependent, a resource that builds upon another can also benefit from controls applied to the underlying resource, ensuring that threats are considered across multiple layers of the environment.

The identified threats in the category *Compute* are summarized in Table 6.1. The entries are grouped by subcategory and ordered by severity.

Subcategory	Overall Severity	Threat Description
compute_block_storage_no_snapshots:		
Block Storage	Medium	Having no recent snapshots may indicate gaps in a reliable backup and recovery strategy.
compute_block_storage_unused_volume:		
Block Storage	Medium	Having unused block storage volumes increases costs without providing actual functionality.
compute_eip_shodan:		
Elastic IP	Critical	Shodan indexes exposed systems and is a potential resource for adversaries to find exploitable systems, which adds attention to the attack surface of resources.
compute_eip_unused:		
Elastic IP	Informational	Having unused elastic IPs increases costs without apparent benefit.
compute_instance_pool_instances_different_type:		
Instance Pools	High	If some instances in an instance pool are not running the specified type, unpatched instances, deprecated services, or incompatibility can be the consequence.
compute_instance_pool_instances_different_template:		
Instance Pools	High	If some instances in an instance pool are not running the specified OS template, unpatched instances, deprecated services, or incompatibility can be the consequence.
compute_instance_pool_instances_state_not_running:		

Continued on next page

Subcategory	Overall Severity	Threat Description
Instance Pools	High	If instances in an instance pool are not in a running state, performance degradation, reduced availability, or inability to handle expected workloads may be the consequence.
compute_instance_public_ingress_all_ports:		
Instances	Critical	The instance might be exposed to the internet on layer four, leading to potential unauthorized access, data disclosure, or DoS.
compute_instance_public_ingress_kerberos_ldap_ports:		
Instances	Critical	The instance is public, and the attached security group allows inbound traffic for Kerberos and LDAP ports, leading to potential unauthorized access.
compute_instance_public_ingress_kibana_and_elastic_ports:		
Instances	Critical	The instance is public, and the attached security group allows inbound traffic for Kibana and Elastic ports, leading to potential unauthorized access.
compute_instance_public_ingress_mgmt_ports:		
Instances	Critical	The instance is public, and the attached security group allows inbound traffic for remote management protocol ports, leading to potential unauthorized access.
compute_instance_public_ingress_mail_ports:		
Instances	Critical	The instance is public, and the attached security group allows inbound traffic for ports associated with mail services, leading to potential unauthorized access.
compute_instance_public_ingress_file_transfer_ports:		

Continued on next page

Subcategory	Overall Severity	Threat Description
Instances	Critical	The instance is public, and the attached security group allows inbound traffic for ports associated with file transfer protocols, leading to potential unauthorized access.
compute_instance_public_ingress_database_ports:		
Instances	Critical	The instance is public, and the attached security group allows inbound traffic for database service ports, leading to potential unauthorized access.
compute_instance_public_ingress_from_quad_zero:		
Instances	High	The instance might be exposed to the internet on layer three with layer four restrictions, leading to potential unauthorized access, data disclosure, or DoS.
compute_instance_majority_in_same_zone:		
Instances	High	If one zone fails, the majority of instances will cease to operate, decreasing resilience.
compute_instance_no_anti_affinity_group:		
Instances	High	If multiple instances depend on the same host, a potential failure will have a larger impact.
compute_instance_older_than_specific_days:		
Instances	High	Old instances might contain outdated or vulnerable software.
compute_instance_outdated_template_used:		
Instances	High	Using deprecated templates can result in the deployment of unpatched software and outdated OS images.
compute_instance_without_snapshot:		
Instances	Medium	Missing snapshots can hinder recovery efforts of failed instances and can cause data loss.
compute_instance_large_instance_type:		

Continued on next page

Subcategory	Overall Severity	Threat Description
Instances	Low	Using large instance types when not necessary increases costs.
compute_instance_unreachable:		
Instances	Informational	The instance is unreachable but still requires resources, raising costs.
compute_load_balancer_security_health_check_fails:		
Load Balancers	High	If health checks fail, the load balancer does not forward traffic to the instances, resulting in reduced availability.
compute_load_balancer_security_group_blocks_health_check:		
Load Balancers	High	If health checks are blocked by security groups, the load balancer does not forward traffic to the instances, resulting in reduced availability.
compute_private_networks_leases_reach_threshold:		
Private Networks	Medium	When the private networks' capacity is reached, deployment failures and potential service interruptions may be the consequence.
compute_sg_allow_ingress_from_quad_zero:		
Security Groups	Medium	Security group allows unrestricted inbound traffic on layer three, leading to potential unauthorized access if attached to resources.
compute_sg_allow_ingress_from_all_ports:		
Security Groups	Medium	Security group allows unrestricted inbound traffic on layer four, leading to potential unauthorized access if attached to resources.
compute_sg_allow_ingress_from_internet_to_database_ports:		
Security Groups	Medium	Security group allows inbound traffic for database service ports, leading to potential unauthorized access if attached to resources.

Continued on next page

Subcategory	Overall Severity	Threat Description
compute_sg_allow_ingress_from_internet_to_kerberos_ldap_ports:		
Security Groups	Medium	Security group allows inbound traffic for Kerberos and LDAP ports, leading to potential unauthorized access if attached to resources.
compute_sg_allow_ingress_from_internet_to_kibana_and_elastic_ports:		
Security Groups	Medium	Security group allows inbound traffic for Kibana and Elastic ports, leading to potential unauthorized access if attached to resources.
compute_sg_allow_ingress_from_internet_to_mgmt_ports:		
Security Groups	Medium	Security group allows inbound traffic for remote management protocol ports, leading to potential unauthorized access if attached to resources.
compute_sg_allow_ingress_from_internet_to_mail_ports:		
Security Groups	Medium	Security group allows inbound traffic for ports associated with mail services, leading to potential unauthorized access if attached to resources.
compute_sg_allow_ingress_from_internet_to_file_transfer_ports:		
Security Groups	Medium	Security group allows inbound traffic for ports associated with file transfer protocols, leading to potential unauthorized access if attached to resources.
compute_sg_allow_ingress_from_prefixes_length_to_16:		
Security Groups	Medium	The security group allows inbound traffic on layer three from IPv4 networks as large as a class B network (approximately 65,000 addresses), leading to potential unauthorized access if attached to resources.
compute_sg_unused_group:		

Continued on next page

Subcategory	Overall Severity	Threat Description
Security Groups	Medium	Having unused security groups can reduce transparency and complicate the management of network policies.
compute_sks_version_outdated:		
SKS	Medium	Using versions that are no longer offered by Exoscale may indicate old resources, increasing the possibility of running vulnerable software.
compute_sks_auto_update_disabled:		
SKS	Medium	By disabling auto-update, the control plane version might become deprecated, which may lead to outdated resources and vulnerabilities.
compute_ssh_keys_older_than_specific_days:		
SSH Keys	Medium	SSH keys could be compromised over time and allow unauthorized access to attached resources.
compute_ssh_keys_unused:		
SSH Keys	Medium	Having unused SSH keys can originate from deprecated configurations or former employees and can induce administrative errors.

Table 6.1: Overview of threats found in the category *Compute*

6.1.2 Implemented Controls in Storage

In the *Storage* category, the analysis focuses on the configuration of buckets, as these represent the central control point for managing access to stored data. Most identified threats relate to the public exposure of buckets, such as unrestricted access policies or no restrictions on cross-origin requests. While individual objects also support ACLs, these are not evaluated in the current approach. Assessing every object would require querying a potentially unrestricted number of items through the API, which does not scale. The defined threats in the category *Storage* are summarized in Table 6.2. The entries are sorted by severity.

Subcategory	Severity	Threat Description
storage_buckets_acl_full_control_AllUsers:		
Object Storage	Critical	Full access carries the threat of disclosure and manipulation of consumer data.
storage_buckets_acl_read_AllUsers:		
Object Storage	Critical	Read access for all users carries the threat of data disclosure.
storage_buckets_acl_read_write_AllUsers:		
Object Storage	Critical	Read and write access for all users carries the threat of disclosure and manipulation of consumer data.
storage_buckets_cors_unrestricted_wildcard:		
Object Storage	Critical	Allowing CORS responses from any origin may expose sensitive data or APIs to untrusted web clients, especially if credentials or sensitive headers are allowed.
storage_buckets_versioning_enabled:		
Object Storage	High	File versioning facilitates the recovery of data from user actions or application failures.

Table 6.2: Overview of threats found in the category *Storage*

6.1.3 Implemented Controls in DBaaS

In the *DBaaS* category, the evaluation focuses on general platform-level configuration options rather than service-specific database settings. Controls that are relevant to multiple database offerings but are provided through the same API calls are grouped in the *General* subcategory. The scope of the analysis is limited to the SQL-based services MySQL and PostgreSQL. The defined threats in the category *DBaaS* are summarized in Table 6.3. The entries are grouped by their respective subcategory and then sorted by severity.

Subcategory	Severity	Threat Description
dbaas_general_termination_protection_off:		
General	High	Having termination protection enabled prevents unintended disruptions to consumer services through termination.
dbaas_mysql_ip_filter_allows_all:		
MySQL	Critical	Having no access restrictions exposes consumer resources to the internet and enables potential unauthorized access.
dbaas_mysql_require_ssl:		
MySQL	High	Using unencrypted protocols to access consumer data may result in data leaks.
dbaas_mysql_unreachable:		
MySQL	Medium	Having unintentionally unreachable databases increases costs.
dbaas_postgresql_ip_filter_allows_all:		
PostgreSQL	Critical	Having no access restrictions exposes consumer resources to the internet and enables potential unauthorized access.
dbaas_postgresql_require_ssl:		
PostgreSQL	High	Using unencrypted protocols to access consumer data may result in data leaks.
dbaas_postgresql_weak_password_hash:		
PostgreSQL	High	Using deprecated `password_encryption` configuration options provides no protection if password hashes are leaked.
dbaas_postgresql_unreachable:		
PostgreSQL	Medium	Having unintentionally unreachable databases increases costs.

Table 6.3: Overview of threats found in the category *DBaaS*

6.1.4 Implemented Controls in IAM

The identified threats in the *IAM* category have the potential for privilege escalation or unintended authorizations. The corresponding controls and their severity levels are summarized in Table 6.4. The entries are grouped by their respective subcategory and then sorted by severity.

Subcategory	Severity	Threat Description
iam_users_mfa_enabled:		
Users	Critical	Unauthorized access to these accounts if the password is not secure or is disclosed in any way.
iam_roles_full_access:		
Roles	Medium	Unintended access to all actions could lead to privilege escalation and unauthorized access to sensitive resources.
iam_roles_iam_full_access:		
Roles	Medium	Unintended access to all IAM actions could lead to privilege escalation and unauthorized access to sensitive resources.

Table 6.4: Overview of threats found in the category *IAM*

6.2 Prototype Results

The developed prototype is compatible with Windows and currently covers the services *Compute*, *Storage*, *IAM*, as well as the MySQL and PostgreSQL offerings within the category *DBaaS*. Its execution flow is based on dynamically traversing the predefined directory structure to locate and load available controls. For each resource type, the prototype issues the necessary API requests, parses the responses into structured objects, and caches them in temporary files. The control logic is then applied to the respective resources, generating findings whenever one of the previously defined potential threats is detected. Findings are reported in a structured format on the CLI, including severity, description, identified threats, and recommendations, ensuring that results are reproducible. This design allows for modular extension, enabling the integration of additional services and controls in the future. An exemplary output is depicted in Figure 6.1.

Figure 6.1: Example output of Exoscan

However, certain controls may detect the same issue in multiple contexts, such as exposed ports at both the network and instance levels, resulting in findings that could appear redundant. Additionally, the tool is inherently limited by the variability of cloud consumer environments. Each consumer may have unique requirements, distinct network configurations, or different operational policies, making it impossible to define a single baseline applicable to all consumers. These context-specific variations impose inherent limitations on the generalizability of the findings. The prototype demonstrates how security implications for cloud resources can be automatically identified and systematically evaluated. The complete source code of the prototype is publicly available on GitHub.[1]

6.3 Discussion

The identified threats and the developed prototype presented in the previous section illustrate the key accomplishments of this thesis. Beyond meeting the defined objectives, the work also revealed insights into both the opportunities and obstacles faced.

[1] https://github.com/NorthernIceman/exoscan

One notable outcome was the implementation of a two-dimensional scoring system for classifying threats. Although not a primary objective, this approach proved valuable, producing severity ratings consistent with those of comparable tools for other cloud providers. Furthermore, the practical verification of the identified threats confirmed that the derived security controls are applicable in real-world scenarios.

However, the scope of the analysis was limited. Only base cloud services were examined, while not all subcategories within the *DBaaS* category could be evaluated. As a result, certain service-specific configurations remain unanalyzed, even though they may also have security implications.

The development of the prototype benefited from the availability of Exoscale API documentation, which made it possible to model resources as Python classes. The two-step approach required to retrieve complete resource information proved effective, and the structured design of the API calls supported a reproducible implementation.

The decision to adopt a modular architecture turned out to be advantageous. It allowed security controls to be added systematically, even though some duplication of entries was accepted as a tradeoff. Overall, the integration of the defined controls into the prototype worked well, and the design ensures that additional services or checks can be incorporated in future iterations.

7 Conclusion

The growing market for cloud infrastructure services underlines that this field continues to evolve and remains relevant, particularly as it forms an essential component of modern IT landscapes. Within the framework of the shared responsibility model, consumers retain accountability for the configuration and management of those resources under their control. While vulnerability and threat scanning tools exist for the three largest cloud infrastructure providers, no equivalent solution is currently available for the European provider Exoscale.

This work addresses that gap by systematically analyzing Exoscale's base cloud services, identifying potential threats, and formulating corresponding security controls in the categories *Compute*, *IAM*, *DBaaS*, and *Storage*. Each identified threat was assessed along the dimensions of its potential impact on confidentiality, integrity, and availability, as well as its degree of exposure. The sum of the individual scores of these assessments resulted in the overall severity classification ranging from `Informational` to `Critical`.

To enable programmatic evaluation of the defined controls, a prototype command-line tool was developed in Python as part of this thesis. The tool issues API calls to enumerate the consumer's existing cloud resources and subsequently applies the control logic associated with each identified threat. In doing so, it demonstrates how security-relevant misconfigurations and weaknesses can be detected automatically in Exoscale environments.

7.1 Future Work

While this thesis demonstrates how security-relevant configurations in Exoscale can be systematically analyzed and evaluated, several directions remain open for further work. These can be grouped into two broad areas: the refinement of threat analysis and the extension of the prototype's functionality.

7.1.1 Threat Analysis

Future iterations could expand the scope of covered services beyond the categories *Compute*, *Storage*, *IAM*, and *DBaaS*. Incorporating additional Exoscale offerings would allow for a more comprehensive assessment of security-relevant configuration options across the platform.
Moreover, the current prototype primarily evaluates base configuration parameters. A possible extension would be the inclusion of contextual factors. For example, application-specific settings like advanced database parameters could be analyzed to capture potential misconfigurations that extend beyond the platform layer.
Another possible enhancement would be to map the identified threats and controls to established security frameworks such as ISO/IEC 27001:2022 Annex A Controls or the MITRE ATT&CK Cloud Matrix[1]. This would provide practitioners with a reference point for compliance and governance requirements, bridging the gap between technical findings and organizational security standards.

7.1.2 Prototype Enhancements

From a technical perspective, the prototype must remain adaptable to changes in Exoscale's services and API. Ongoing maintenance will be required to ensure compatibility with changed endpoints and newly introduced configuration options.
In addition, extending the range of output formats beyond the command line could improve usability. Exporting results into structured reports would support integration into existing workflows and security dashboards. Cross-platform compatibility is another key improvement, particularly adding support for Linux environments alongside the current Windows version.
Finally, further functionality could be introduced, such as generating a complete inventory of a consumer's Exoscale resources.

[1] https://attack.mitre.org/matrices/enterprise/cloud/

List of Tables

List of Listings

Acronyms

ACL	Access Control Lists
ACP	Access Control Policy
AH	Authentication Header
API	Application Programming Interface
ARP	Address Resolution Protocol
BSI	Federal Office for Information Security
CA	Certificate Authority
CCM	Cloud Controller Manager
CDN	Content Delivery Network
CIA	Confidentiality, Integrity, and Availability
CIDR	Classless Inter-Domain Routing
CLI	Command-Line Interface
CNI	Container Network Interface
CORS	Cross-Origin Resource Sharing
CSI	Container Storage Interface
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
DBaaS	Database as a Service
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DoS	Denial of Service

ESP	Encapsulating Security Payload
GRE	Generic Routing Encapsulation
HTTP	Hypertext Transfer Protocol
HTTPs	Hypertext Transfer Protocol Secure
IaaS	Infrastructure as a Service
IAM	Identity and Access Management
ICMP	Internet Control Message Protocol
ICMPv6	Internet Control Message Protocol for the Internet Protocol Version 6
IPIP	IP-in-IP
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
JSON	JavaScript Object Notation
KVM	Kernel-based Virtual Machine
LDAP	Lightweight Directory Access Protocol
MAC	Media Access Control
MFA	Multi Factor Authentication
PaaS	Platform as a Service
RDP	Remote Desktop Protocol

SaaS	Software as a Service
SKS	Scalable Kubernetes Service
SLA	Service Level Agreement
SOS	Simple Object Storage
SQL	Structured Query Language
SSE-C	Server-Side Encryption by Customer-Provided Keys
SSH	Secure Shell
SSL	Secure Sockets Layer
SSO	Single Sign-On
TCP	Transmission Control Protocol
TPM	Trusted Platform Module
UDP	User Datagram Protocol
URL	Uniform Resource Locator
VLAN	Virtual Local Area Network

Bibliography

[1] MordorIntelligence, *Cloud computing market size and share*, 2024. [Online]. Available: https://www.mordorintelligence.com/industry-reports/cloud-computing-market (visited on 08/07/2025).

[2] G. Amudha, P. Gopika, Saranya G, Vinoth R, M.G. Dinesh, and Siva Subramanian R, "Cloud computing: Transformations, opportunities, and challenges," in *2025 3rd International Conference on Self Sustainable Artificial Intelligence Systems (ICSSAS)*, 2025, pp. 897–903. DOI: 10.1109/ICSSAS66150.2025.11080697.

[3] Felix Richter, *The Big Three Stay Ahead in Ever-Growing Cloud Market*, 2025. [Online]. Available: https://www.statista.com/chart/18819/worldwide-market-share-of-leading-cloud-infrastructure-service-providers/ (visited on 09/07/2025).

[4] Peter Mell and Timothy Grance, *The NIST Definition of Cloud Computing*, 2011. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-145.pdf (visited on 09/07/2025).

[5] National Cyber Security Centre, *Cloud security shared responsibility model*, 2022. [Online]. Available: https://www.ncsc.gov.uk/collection/cloud/understanding-cloud-services/cloud-security-shared-responsibility-model (visited on 09/07/2025).

[6] Clarisse Mendoza and Cristina Reyes, "Exploring the impact of shared responsibility models on cloud security posture and vulnerability management," *Journal of Emerging Technologies*, Apr. 2023.

[7] Santanu Koley, Jayeeta Majumdar, Mihir Baran Bera, and Pinaki Pratim Acharjya, "Multitenancy architecture for augmented security in cloud computing," in *2023 5th International*

Conference on Inventive Research in Computing Applications (ICIRCA), 2023, pp. 1090–1094. DOI: 10.1109/ICIRCA57980.2023.10220638.

[8] Cloudflare, *What ia a cloud API*, 2025. [Online]. Available: https://www.cloudflare.com/learning/security/api/what-is-a-cloud-api/ (visited on 09/07/2025).

[9] Cloud Security Alliance, *[top] Threats to Cloud Computing 2024*, 2024. [Online]. Available: https://cloudsecurityalliance.org/artifacts/top-threats-to-cloud-computing-2024 (visited on 09/07/2025).

[10] Exoscale, *Zones – Exoscale Documentation*, Documentation, 2025. [Online]. Available: https://community.exoscale.com/platform/dc-zones/ (visited on 08/11/2025).

[11] Exoscale, *Compute – Exoscale Documentation*, Documentation, 2025. [Online]. Available: https://community.exoscale.com/product/compute/instances/overview/ (visited on 08/11/2025).

[12] Exoscale, *Anti-Affinity Groups – Exoscale Documentation*, Documentation, 2025. [Online]. Available: https://community.exoscale.com/product/compute/instances/how-to/anti-affinity/ (visited on 08/11/2025).

[13] Exoscale, *Block-Storage – Exoscale Documentation*, Documentation, 2025. [Online]. Available: https://community.exoscale.com/product/storage/block-storage/overview/ (visited on 08/11/2025).

[14] Exoscale, *Elastic IP – Exoscale Documentation*, Documentation, 2025. [Online]. Available: https://community.exoscale.com/product/networking/eip/overview/ (visited on 08/11/2025).

[15] Exoscale, *Instance Pools – Exoscale Documentation*, Documentation, 2025. [Online]. Available: https://community.exoscale.com/product/compute/instances/how-to/instance-pools/ (visited on 08/11/2025).

[16] Exoscale, *Load Balancers – Exoscale Documentation*, Documentation, 2025. [Online]. Available: https://community.exoscale.com/product/networking/nlb/overview/ (visited on 08/11/2025).

[17] Exoscale, *Private Networks – Exoscale Documentation*, Documentation, 2025. [Online]. Available: https://community.exoscale.com/product/networking/private-network/overview/ (visited on 08/11/2025).

[18] Exoscale, *Security Groups – Exoscale Documentation*, Documentation, 2025. [Online]. Available: https://community.exoscale.com/product/networking/security-group/overview/ (visited on 08/11/2025).

[19] Exoscale, *SKS – Exoscale Documentation*, Documentation, 2025. [Online]. Available: https://community.exoscale.com/product/compute/containers/overview/ (visited on 08/11/2025).

[20] Exoscale, *SSH Keys – Exoscale Documentation*, Documentation, 2025. [Online]. Available: https://community.exoscale.com/product/compute/instances/how-to/ssh-keypairs/ (visited on 08/11/2025).

[21] Exoscale, *Custom Templates – Exoscale Documentation*, Documentation, 2025. [Online]. Available: https://community.exoscale.com/product/compute/instances/how-to/custom-templates/ (visited on 08/11/2025).

[22] Exoscale, *Object Storage – Exoscale Documentation*, Documentation, 2025. [Online]. Available: https://community.exoscale.com/product/storage/object-storage/overview/ (visited on 08/11/2025).

[23] Exoscale, *Storage Encryption – Exoscale Documentation*, Documentation, 2025. [Online]. Available: https://community.exoscale.com/product/storage/object-storage/how-to/encryption/ (visited on 08/11/2025).

[24] Exoscale, *DBaaS – Exoscale Documentation*, Documentation, 2025. [Online]. Available: https://community.exoscale.com/product/dbaas/overview/ (visited on 08/11/2025).

[25] Exoscale, *MySQL – Exoscale Documentation*, Documentation, 2025. [Online]. Available: https://community.exoscale.com/product/dbaas/service-specific/mysql/overview/ (visited on 08/11/2025).

[26] Exoscale, *PostgreSQL – Exoscale Documentation*, Documentation, 2025. [Online]. Available: https://community.exoscale.com/product/dbaas/service-specific/postgresql/ (visited on 08/11/2025).

[27] Exoscale, *Exoscale IAM*, 2025. [Online]. Available: https://community.exoscale.com/product/iam/overview/ (visited on 08/07/2025).

[28] Exoscale, *Roles and Policies – Exoscale Documentation*, Documentation, 2025. [Online]. Available: https://community.exoscale.com/product/iam/operation/roles-policies/ (visited on 08/11/2025).

[29] Exoscale, *Api Keys and Users – Exoscale Documentation*, Documentation, 2025. [Online]. Available: https://community.exoscale.com/product/iam/operation/users-keys/ (visited on 08/11/2025).

[30] Exoscale, *Exoscale DNS*, 2025. [Online]. Available: https://community.exoscale.com/product/networking/dns/quick-start/ (visited on 08/07/2025).

[31] Longhu Cao and Dan Liu, "Security scanner system of ovirt cloud platform," in *2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, 2016, pp. 357–360. DOI: 10.1109/ICSESS.2016.7883085.

[32] FNU Jimmy, "Cloud security posture management: Tools and techniques," *Journal of Knowledge Learning and Science Technology ISSN: 2959-6386 (online)*, vol. 2, Nov. 2023. DOI: 10.60087/jklst.vol2.n3.p622.

[33] Rajashekar Reddy Yasani, Putalpattu Muni Prasad, Pattlola Srinivas, N V Raja Sekhar Reddy, Parag Jawarkar, and Vedaprada Raghunath, "Ai-driven solutions for cloud security implementing intelligent threat detection and mitigation strategies," in *2024 International Conference on Intelligent Computing and Emerging Communication Technologies (ICEC)*, 2024, pp. 1–6. DOI: 10.1109/ICEC59683.2024.10837032.

[34] Ionel Gordin, Adrian Graur, Alin Potorac, and Doru Balan, "Security assessment of openstack cloud using outside and inside software tools," in *2018 International Conference on Development and Application Systems (DAS)*, 2018, pp. 170–174. DOI: 10.1109/DAAS.2018.8396091.

[35] Kazi Wali Ullah, Abu Shohel Ahmed, and Jukka Ylitalo, "Towards building an automated security compliance tool for the cloud," in *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, 2013, pp. 1587–1593. DOI: 10.1109/TrustCom.2013.195.

[36] Oumayma Mejri, Dana Yang, and Inshil Doh, "Cloud security issues and log-based proactive strategy," in *2021 23rd International Conference on Advanced Communication Technology (ICACT)*, 2021, pp. 392–397. DOI: 10.23919/ICACT51234.2021.9370392.

[37] Pawan Kumar Goel and Amit Singhal, “Security issues and threats in cloud computing: Problems and solutions,” in *2023 3rd International Conference on Advancement in Electronics & Communication Engineering (AECE)*, 2023, pp. 1019–1023. DOI: 10.1109/AECE59614.2023.10428390.

[38] C Jansi Sophia Mary, K Mahalakshmi, and B Senthilkumar, “Deep dive on various security challenges, threats and attacks over the cloud security,” in *2023 9th International Conference on Advanced Computing and Communication Systems (ICACCS)*, vol. 1, 2023, pp. 2089–2094. DOI: 10.1109/ICACCS57279.2023.10113125.

[39] Deneesh Narayanasamy, “Cloud security 101: Understanding shared responsibility and securing your data,” *INTERNATIONAL JOURNAL OF COMPUTER ENGINEERING & TECHNOLOGY*, vol. 16, pp. 704–716, Jan. 2025. DOI: 10.34218/IJCET_16_01_058.

[40] Exoscale, *Services – Exoscale Documentation*, Documentation, 2025. [Online]. Available: https://community.exoscale.com/platform/exoscale/ (visited on 08/11/2025).

[41] Exoscale, *Cloud Compute – Scalable Virtual Machines and Cloud Servers in Europe | Exoscale*, 2025. [Online]. Available: https://www.exoscale.com/compute/ (visited on 08/07/2025).

[42] Exoscale, *S3 Object Storage as a Service - Secure in Europe | Exoscale*, 2025. [Online]. Available: https://www.exoscale.com/object-storage/ (visited on 08/07/2025).

[43] Exoscale, *Managed Database as a Service (DBaaS) in Europe | Exoscale*, 2025. [Online]. Available: https://www.exoscale.com/dbaas/ (visited on 08/06/2025).

[44] stackoverflow, *Most popular technologies Databases*, 2024. [Online]. Available: https://survey.stackoverflow.co/2024/technology#most-popular-technologies-database (visited on 08/07/2025).

[45] Exoscale, *DNS Hosting in Europe | Exoscale*, 2025. [Online]. Available: https://www.exoscale.com/dns/ (visited on 08/07/2025).

[46] Exoscale, *Exoscale Marketplace*, 2025. [Online]. Available: https://www.exoscale.com/marketplace/ (visited on 08/07/2025).

[47] Exoscale, *Password Authentication – PostgreSQL Documentation*, Documentation, 2025. [Online]. Available: https://www.postgresql.org/docs/17/auth-password.html (visited on 08/11/2025).

[48] Suganya Subramani, A. R. Kavitha, and A. Rukshana Safrin, “Zero trust network architecture for modern enterprise environments,” in *2025 International Conference on Data Science, Agents & Artificial Intelligence (ICDSAAI)*, 2025, pp. 1–6. DOI: 10.1109/ICDSAAI65575.2025.11011897.

[49] Exoscale, *Exoscale API*, 2025. [Online]. Available: https://openapi-v2.exoscale.com/ (visited on 08/07/2025).