ARCTIC OWL

Report

Confidential

Penetration Test

StackHPC LTD

ARCTIC OWL CYBER SECURITY LTD

Ardfinnan House, 17 Trinity Street

Dublin 2, D02W281, Ireland

www.arcticowl.se

hello@arcticowl.se

Executive Summary

Arctic Owl Cyber Security LTD carreied out a penetration test to evaluate the security posture for the deployed Azimith and Zentih applications between December 11, 2024, and February 13, 2025, focusing on three key areas: the web application, architecture, and Kubernetes infrastructure.

The assessment confirmed a robust security posture, with no high or critical vulnerabilities identified that require immediate attention. Consistent hardening practices were noted throughout the project, deployment, etc. Strong points include attack surface reduction, authentication mechanisms, and multi-layered security configuration.

Recommendations

- Prioritize Remediation: Address identified vulnerabilities, particularly focusing on updating the keycloak if any clients are using LDAP.
- Continue to Adopt Layered Security: Continue to implement security hardening measures across all levels of the application and infrastructure.
- Integrate Automations: Continue the automations for identifying and updating known vulnerabilities in third-party dependencies.
- Patch Management Process: Review and fromalize patch management process for libraries and container images identified by automation.

Conclusion

The project have a good secure posture with some identified weakpoints and possible hardening surgestions. Some of the identified vulnerabilities could have been more severe but good manual pratices, attack surface mangement and secure configurations have reduced the risk significantly. By addressing the identified vulnerabilities and implementing the recommendations, the project will be in a even better position maintain a strong security posture.

ARCTIC OWL

Contents

1 Introduction

1.1 Engagement Contacts

Name	Role	Email
Erik Wilhelmsson	Cyber Security Consultant	erik@arcticowl.se

1.2 Revision History

Version	Date	Author	Description
0.1	2024-12-17		Document creation
0.2	2025-02-11		First draft
1.0	2025-02-13		Presentation draft
1.1	2025-02-18		Released
1.2	2025-02-18		OpenStack related findings and recommendations removed

1.3 Overview

This project was carried out from the mid of December 2024 to mid-February 2025. Arctic Owl Cyber Security was contracted to conduct a comprehensive penetration test of the Azimuth/Zenith open-source projects, which consists of a platform to help clients manage their OpenStack environments.

1.4 Purpose and Scope

This project aims to identify vulnerabilities, misconfiguration, and hardening suggestions within the defined scope to improve the security posture of the application and underlying infrastructure. The test environment `portal.apps.195-114-30-100.sslip.io` was used throughout the project to simulate attack scenarios and evaluate the system's security posture.

GitHub projects analyzed during the project:

- Main components:
  - https://github.com/azimuth-cloud/zenith
  - https://github.com/azimuth-cloud/azimuth
- Sub components:
  - https://github.com/azimuth-cloud/azimuth-identity-operator
  - https://github.com/azimuth-cloud/azimuth-caas-operator
  - https://github.com/azimuth-cloud/azimuth-llm
  - https://github.com/azimuth-cloud/azimuth-images

Focus Areas:

- Common web application vulnerabilities including but not limited to OWASP top 10
- Components used for Authentication, Authorization, and Session Management
- Third-party dependencies used in the source code
- Third-party dependencies in deployment tools and containers such as Ansible, OpenTofu, Dockerfiles
- Reviewing the architecture of the SSH reverse tunneling feature
- Core components such as operators for Kubernetes, Jupyter Notebook, and Keycloak
- Apache Guacamole and Grafana interfaces

The following were not included in the scope:

- A full code review of the source code.

1.5 Execution

Our penetration testing methodology for this project adopted a dynamic and layered approach, focusing on gaining a deep understanding of the target environment to identify realistic attack scenarios. Our efforts can be broken up into three key domains:

- Web Application Testing: A thorough code-supported testing approach was employed to identify potential vulnerabilities in the web application, this included manual and automated techniques.
- Architecture Review : Critical features of the application were analyzed for vulnerabilities, focusing on identifying architectural weakpoints and potential attack vectors.
- Kubernetes Review : The Kubernetes infrastructure was reviewed to ensure secure configurations, proper access controls, and resilience against exploitation attackers and lateral movement from unauthorized users.

2 Findings

This section contains the vulnerabilities found in the environment of the customer.

2.1 Insufficient Output Encoding (Informational)

During the assignment, the testers identified a potential cross-site scripting vulnerability due to lacking output controls in a templating engine initated in the following location:
`azimuth/cluster_engine/engine.py`[1]

Currently the output will bypass HTML escaping, exposing the application to cross-site scripting (XSS) vulnerabilities. However, there is no feature to modify or add new objects that will be managed by the template.

Recommendations

Document the use of Jinja2 in the application and ensure that user input is properly validated and the outputs are properly encoded with context in mind before using this component with user provided data. `Cross Site Scripting Prevention Cheat Sheet`[2]

[1] https://github.com/azimuth-cloud/azimuth/blob/1c5b37f8683e8017a4240c30bd698810ecb30763/api/azimuth/cluster_engine/engine.py#L62

[2] https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

2.2 Adding Security.md to the Repositories (Informational)

When looking at the repositories of the project, we noticed that there is no `SECURITY.md` file in any of the repositories. It's considered good pratice to have clear information how to report security vulnerabilities to the project.

Recommendations

Add a `SECURITY.md` file to the root of all the repository to provide a clear and concise process for reporting security vulnerabilities. Here is an example of a generic that could be used:

```
# Security Policy

## Reporting a Vulnerability

Thank you for taking the time to improve the security of this project. We
↪  take security issues seriously and appreciate your responsible
↪  disclosure.

If you believe you have found a security vulnerability in this repository,
↪  please use GitHub's built-in **"Report a vulnerability"** feature to
↪  notify us privately:

1. Navigate to the **Security** tab at the top of this repository.
2. Click on the **"Report a vulnerability"** button.
3. Fill out the form with details about the vulnerability and submit it.

This ensures that only repository maintainers and authorized personnel can
↪  view the report.

### What to Include in Your Report
To help us address the issue effectively, please include:
- A clear and detailed description of the vulnerability.
- Steps to reproduce the issue.
- Any potential impact of the vulnerability.
- Suggestions for mitigation, if possible.

### Response Time
We are committed to investigating and responding to reported
↪  vulnerabilities promptly. You can expect:
- An acknowledgment of your report within 48 hours.
- Updates as we progress on resolving the issue.
- Notification when the issue is resolved.

Thank you for helping us keep this project secure!
```

Figure 1: More information about the `SECURITY.md` file can be found in the GitHub documentation[3].

2.3 Keycloak serverinfo Endpoint Exposure (Informational)

The Keycloak `serverinfo` endpoint exposes detailed server configuration information, including server version, installed modules, and runtime environment details. This endpoint is intended for administrative use, but if improperly secured, it may be accessible to unauthorized users. Exposing this endpoint to non-administrative users or systems increases the risk of sensitive information disclosure, which attackers could use to identify vulnerabilities or target specific components of the environment.

```
HTTP/2 200 OK
Date: Mon, 16 Dec 2024 11:16:13 GMT
Content-Type: application/json;charset=UTF-8
Cache-Control: no-cache
Referrer-Policy: no-referrer
Strict-Transport-Security: max-age=31536000; includeSubDomains
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-Xss-Protection: 1; mode=block

{"systemInfo":{"version":"25.0.6","serverTime":"Mon Dec 16 11:16:13 GMT
↪  2024",
"uptime":"6 days, 1 hour, 37 minutes, 20 seconds","uptimeMillis":524240309,
"javaVersion":"21.0.4","javaVendor":"Red Hat, Inc.","javaVm":"OpenJDK
↪  64-Bit Server VM",
"javaVmVersion":"21.0.4+7-LTS","javaRuntime":"OpenJDK Runtime Environment",
"javaHome":"/usr/lib/jvm/java-21-openjdk-21.0.4.0.7-1.el9_⌋
↪  .x86_64","osName":"Linux",
"osArchitecture":"amd64","osVersion":"5.15.0-126-_⌋
↪  generic","fileEncoding":"UTF-8",
"userName":"keycloak","userDir":"/_⌋
↪  ","userTimezone":"GMT","userLocale":"en_US"},
"memoryInfo":{"total":1503657984,"totalFormated":"1434
↪  MB","used":111079128,
"usedFormated":"105
↪  MB","free":1392578856,"freePercentage":92,"freeFormated":"1328 MB"},
"profileInfo":{"name":"default","disabledFeatures":[_⌋
↪  "ADMIN_FINE_GRAINED_AUTHZ"]
...250KB similar data...}}
```

Recommendations

Best practices dictate that the `serverinfo` endpoint should be accessible only to Keycloak administrators or completely disabled unless required for operational purposes.

- Disable the `serverinfo` endpoint entirely if it is not operationally required.
- Restrict access to the `serverinfo` endpoint strictly to Keycloak administrators using role-based access controls.
- Use network controls, such as firewalls or API gateways, to block access to the `serverinfo` endpoint from untrusted networks.

2.4 Update Practices for Keycloak (Vulnerability - Medium)

During our enagemnet an outdated version (25.0.6) of Keycloak was deployed for the test environment of the project.

This indicates that the current strategy to ensure the Keycloak instance remains consistently up to date requires a review.

Keycloak, as an open-source identity and access management solution, is not reciving any security updates unless licencing is aquired with Red Hat.

```
HTTP/2 200 OK
Date: Mon, 16 Dec 2024 11:16:13 GMT
Content-Type: application/json;charset=UTF-8
Cache-Control: no-cache
Referrer-Policy: no-referrer
Strict-Transport-Security: max-age=31536000; includeSubDomains
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-Xss-Protection: 1; mode=block

{"systemInfo":{"version":"25.0.6",{...}}}
```

Recommendations

The current version of Keycloak is outdated and needs to be updated. However it's not vulnerable to `Keycloak 25.0.5 privilege escalation`[4].

During the Engagement another vulnerability published for Keycloak, `CVE-2025-0604`[5]. When an Active Directory user resets their password, the system updates it without performing an LDAP bind to validate the new credentials against AD. This vulnerability allows users whose AD accounts are expired or disabled to regain access in Keycloak, bypassing AD restrictions. The issue enables authentication bypass and could allow unauthorized access under certain conditions.

- Treat all old verisons of Keycloak as end of life and stay on the latest version.
- Review subscribing to Red Hat's supported version of Keycloak to access backported patches and vendor support if needed.
- Actively track vulnerability announcements for Keycloak and integrate updates into a CI/CD pipeline for streamlined deployment.

[4] https://www.cve.org/CVERecord?id=CVE-2024-8698
[5] https://www.cve.org/CVERecord?id=CVE-2025-0604

- If you have clients using the LDAP integration, stay on top on news about a fix for `CVE-2025-0604` and apply the fix as soon as possible.

2.5 Dependencies With Known Vulnerabilities (Vulnerability - Medium)

When using third-party libraries, it is essential to have a stratergy to keep them updated and to mitigate known vulnerabilities.

The presence of outdated and dependencies affected by known vulnerabilities across all of the repositories indicates a lack of efficient dependency management.

The following libraries are examples of third-party dependencies affected by known vulnerabilities:

azimuth-cloud/zenith

Library	Vulnerability	Fixed in
`aiohttp 3.9.5`	**CVE-2024-52304** (Severity: Moderate): HTTP Request/Response Smuggling	3.10.11
`aiohttp 3.9.5`	**CVE-2024-42367** (Severity: Moderate): UNIX Symbolic Link (Symlink) Following	3.10.2
`cryptography 42.0.8`	**GHSA-h4gh-qq45-vh27** (Severity: Moderate): Vulnerable third-party component (OpenSSL)	43.0.1
`cryptography 43.0.0`	**GHSA-h4gh-qq45-vh27** (Severity: Moderate): Vulnerable third-party component (OpenSSL)	43.0.1
`cryptography 42.0.8`	**GHSA-h4gh-qq45-vh27** (Severity: Moderate): Vulnerable third-party component (OpenSSL)	43.0.1

Table 1: Dependencies found in operator/requirements.txt, sync/requirements.txt, registrar/requirements.txt, sshd/requirements.txt

azimuth-cloud/azimuth

Library	Vulnerability	Fixed in
`babel-plugin -polyfill-corejs2`	**CVE-2023-45133**: Incomplete List Of Disallowed Inputs. Potentially results in arbitrary code execution during compilation.	0.4.6

cookie	**CVE-2024-47764**: Improper Neutralization of Special Elements in Output Used by a Downstream Component ('Injection').	0.7.0
nanoid	**CVE-2024-55565**: Loop with Unreachable Exit Condition ('Infinite Loop').	3.3.8, 5.0.9
path-to-regexp	**CVE-2024-52798**: Inefficient Regular Expression Complexity.	0.1.12
send	**CVE-2024-43799**: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting').	0.19.0
json5	**CVE-2022-46175**: Prototype Pollution in the parse method.	1.0.2, 2.2.2
loader-utils	**CVE-2022-37603**: Regular expression denial of service (ReDoS) flaw in interpolateName function.	1.4.2, 2.0.4, 3.2.1
loader-utils	**CVE-2022-37599**: Uncontrolled Resource Consumption via a regular expression denial of service (ReDoS) flaw.	1.4.2, 2.0.4, 3.2.1
luxon	**CVE-2023-22467**: Inefficient Regular Expression Complexity.	1.28.1, 2.5.2, 3.2.1

Table 2: Dependencies found in ui/yarn.lock

azimuth-cloud/azimuth-caas-operator

Library	Vulnerability	Fixed in
aiohttp 3.9.5	**CVE-2024-52304**: Inconsistent Interpretation of HTTP Requests ('HTTP Request/Response Smuggling').	3.10.11
aiohttp 3.9.5	**CVE-2024-42367**: UNIX Symbolic Link (Symlink) Following.	3.10.2
cryptography 42.0.8	**GHSA-h4gh-qq45-vh27**: Dependency on Vulnerable Third-Party Component.	43.0.1

Table 3: Dependencies found in requirements.txt

azimuth-cloud/azimuth-identity-operator

Library	Vulnerability	Fixed in
`aiohttp 3.9.5`	**CVE-2024-52304**: Inconsistent Interpretation of HTTP Requests ('HTTP Request/Response Smuggling').	3.10.11
`aiohttp 3.9.5`	**CVE-2024-42367**: UNIX Symbolic Link (Symlink) Following.	3.10.2

Table 4: Dependencies found in requirements.txt

stackhpc/azimuth-llm

Library	Vulnerability	Fixed in
`idna`	**CVE-2024-3651**: Uncontrolled Resource Consumption.	3.7
`requests`	**CVE-2024-35195**: Always-Incorrect Control Flow Implementation.	2.32.0;
`tqdm`	**CVE-2024-34062**: Improper Neutralization of Special Elements in Output Used by a Downstream Component ('Injection').	4.66.3
`urllib3`	**CVE-2024-37891**: Incorrect Resource Transfer Between Spheres.	1.26.19, 2.2.2
`certifi`	**CVE-2024-39689**: Insufficient Verification of Data Authenticity.	2024.07.04

Table 5: Dependencies found in scripts/perf-test/requirements.txt

Recommendations

Currently, no vulnerabilities pose a high risk to the system, but it is important to keep dependencies updated to avoid future vulnerabilities. Implement a robust dependency management strategy to identify and address vulnerabilities in third-party dependencies.

2.6 Zenith Use of SSH

In this component, Secure Shell (SSH) is used to establish a reverse tunnel between the client and server components to expose services for clients that do not have an exposed network interface. Examples include web interfaces to administer workloads for compute nodes and the Guacamole user interface for a remote workstation. During the assessment, no weaknesses or hardening measures could be exploited, but several opportunities to improvement were identified.

2.6.1 Reduce Attack Surface by Replacing Secure Shell (Hardening - Low)

Using SSH solely for forwarding ports of various services exposes more features than needed, as it is a full-fledged remote administration tool that can perform remote commands, mount hard disks, or perform file transfers. From an architectural standpoint, the principle of least privilege would advise using a simpler protocol that only deals with network and port forwarding. If we remove the filesystem and command aspects, we reduce the attack vector for the project. Consider using a dedicate VPN solution such as WireGuard.

2.6.2 Client Not Validating Server (Vulnerability - Low)

Certificate verification has been explicitly disabled. This permits insecure connections to insecure servers. Re-enable certificate validation. In specific cases, this could be used to trick the client into connecting to a compromised server. `KnownHosts not checked or added to.`[6]

2.6.3 Reduce Privilege for Port Forward User (Vulnerability - Mid)

When using SSH for port forwarding, it is good practice to set the user's shell to `/usr/sbin/nologin` or `/bin/false` to prevent the user from executing commands.

```
1 sudo usermod -s /usr/sbin/nologin zenith
```

[6] https://github.com/stackhpc/zenith/blob/1a1c67aa6366c4aa458747dfa102cb981b52eea4/client/zenith/client/tunnel.py#L192

2.6.4 Improve sshd_config (Hardening - Mid)

Several hardeningening steps are taken in the configuration of the server's SSH daemon. In addition, more steps can be taken to lock it down even further.

- `PermitTTY No`
  Disable terminal access altogether.
- `PermitOpen 127.0.0.1:1234`
  Limit what ports can be forwarded.
- `ForceCommand`
  Should also limit the user from executing commands.

2.6.5 Improve Key Length for RSA (Hardening - Mid)

In the current setup, the system is using 2048-bit RSA keys, which are not vulnerable to real-world attacks as of yet. When RSA is used, the key length recommended for long-term protection and secure applications is 4096-bit.

2.6.6 Replace RSA (Hardening - Mid)

SSH has support for other stronger encryption algorithems such as Ed25519, and a transition over should be smooth. It is worth pointing out that Elliptic Curve Cryptography (ECC) is not quantum-resistant but more efficient and has better resistance to brute force attacks with shorter key lengths.

2.6.7 Trusting System Link to SSH Binary (Vulnerability - Low)

By using the built-in system binary in the Docker template, this risk can be managed from a binary integrity standpoint. However, code execution in the same context as the SSH daemon can change the binary by adding a malicious binary called `ssh` to `tmpfs` (`/tmp/`) and exporting this into your path variable, allowing for persistence in the container despite having a read-only filesystem without raising privileges or compromising the Docker repository.

As we have a static Docker container, we could just provide a full path for the SSH binary.

```
1 ssh_executable = "/usr/bin/ssh"
```

2.7 Insecure Certificate Validation in SSH Module (Vulnerability - Low)

The `sshd/zenith/sshd/tunnel.py`[7] file contains a Python `requests` call with certificate verification explicitly disabled. This allows insecure connections to unverified servers, exposing the system to potential Man-in-the-Middle attacks.

Recommendations

Review why the validation is disabled and if possible re-enable it.

[7] https://github.com/stackhpc/zenith/blob/1a1c67aa6366c4aa458747dfa102cb981b52eea4/sshd/zenith/sshd/tunnel.py#L105

2.8 Kubernetes Containers with Host Network Access (Vulnerability - Low)

During our engagement, we identified several containers with access to the host network.

In some cases, granting containers access to the host network is necessary to perform specific network operations or to achieve certain performance requirements. However, allowing containers to use the host network is a security risk because it exposes the host's network stack to the container, potentially leading to network-based attacks.

Name	Namespace
kube-controller-manager	kube-system
kube-proxy	kube-system
node-exporter	monitoring-system
node-driver-registrar	openstack-system
calico-node	calico-system
etcd	kube-system
calico-typha	calico-system
liveness-probe	openstack-system
openstack-cloud-controller-manager	openstack-system
kube-proxy	kube-system
tigera-operator	tigera-operator
calico-typha	calico-system
calico-node	calico-system
node-exporter	monitoring-system
node-exporter	monitoring-system
kube-apiserver	kube-system
node-driver-registrar	openstack-system
node-driver-registrar	openstack-system
kube-scheduler	kube-system
liveness-probe	openstack-system
liveness-probe	openstack-system
cinder-csi-plugin	openstack-system
calico-node	calico-system
kube-proxy	kube-system
cinder-csi-plugin	openstack-system
cinder-csi-plugin	openstack-system

Table 6: Containers with Host Network Access

Recommendations

Review if granting containers access to the host network is necessary or if it can be configured with more restrictive network policies.

2.9 Kubernetes Containers Running as Privileged Containers (Vulnerability - Low)

During our engagement, we identified several containers running as privileged containers.

In some cases, running containers as privileged is necessary to perform operations that require elevated permissions, such as accessing host devices, modifying kernel parameters, or accessing hardware such as GPUs. However, running containers as privileged is a security risk because it grants the container extensive access to the host system.

Name	Namespace
kube-api-access-wmrf5	azimuth
kube-api-access-jmgcz	azimuth
kube-api-access-xvfcp	azimuth
kube-api-access-dhk6v	az-stackhpc-blob
kube-api-access-mqbnc	azimuth
kube-api-access-d2jqk	capi-janitor-system
kube-api-access-69cqs	azimuth
kube-api-access-hf849	capo-system
kube-api-access-7g88m	azimuth
kube-api-access-n5q7l	keycloak-system
kube-api-access-nlnls	kube-system
root	monitoring-system
kube-api-access-n6fcv	monitoring-system
kube-api-access-8l84s	capi-system
log	node-problem-detector
kube-api-access-xhnbd	cert-manager
kube-api-access-qj75d	keycloak-system
pgbackrest-config	keycloak-system
usr-local-share-ca-certificates	kube-system
kube-api-access-wn24l	calico-system
etc-ca-certificates	kube-system
kube-api-access-lds6w	kubernetes-dashboard
usr-local-share-ca-certificates	kube-system
lib-modules	calico-system
kube-api-access-n8665	az-stackhpc-ci
kube-api-access-rfwbm	monitoring-system
kube-api-access-j7gnq	monitoring-system
kube-api-access-rrgbw	azimuth
proc	monitoring-system
kube-api-access-pznzn	monitoring-system

Table 7: List of Kubernetes containers running as root (1 of 4)

Name	Namespace
kube-api-access-tr2nn	capi-kubeadm-control-plane-system
kube-api-access-klrwc	calico-apiserver
kube-api-access-zsv8q	azimuth
kube-api-access-tdvhr	monitoring-system
pgbackrest-config	keycloak-system
kube-api-access-v7f2j	calico-system
kube-api-access-2vfvt	openstack-system
pgbackrest-server	keycloak-system
pods	monitoring-system
kube-api-access-gdm7s	kube-system
kube-api-access-b56lw	calico-system
kube-api-access-2vfvt	openstack-system
kube-api-access-mdwjv	openstack-system
k8s-certs	kube-system
kube-api-access-tncpz	openstack-system
kube-api-access-xtjwz	az-stackhpc-azimuth
var-lib-calico	tigera-operator
lib-modules	kube-system
kube-api-access-lwl5t	azimuth
patroni-config	keycloak-system
kubeconfig	kube-system
kube-api-access-l7r5r	calico-apiserver
pgbackrest-config	keycloak-system
kube-api-access-qj75d	keycloak-system
journal	monitoring-system
lib-modules	calico-system
admission-configuration	kube-system
kube-api-access-qj75d	keycloak-system
kube-api-access-sctrm	calico-system
lib-modules	kube-system
kube-api-access-f8d7s	az-stackhpc-dev
machine-id	monitoring-system
kubeconfig	kube-system
kube-api-access-vd4hc	kube-system
kube-api-access-x9dl7	az-stackhpc-lab
machine-id	monitoring-system
kube-api-access-kxhlq	ingress-nginx
kube-api-access-5n97r	kube-system
kube-api-access-x79hl	azimuth
localtime	node-problem-detector

Table 8: List of Kubernetes containers running as root (2 of 4)

Name	Namespace
cert-volume	keycloak-system
kube-api-access-qkpmb	kube-system
lib-modules	kube-system
kube-api-access-c86xj	azimuth
kube-api-access-2vfvt	openstack-system
kube-api-access-qj75d	keycloak-system
kube-api-access-rfwbm	monitoring-system
pods	monitoring-system
kube-api-access-lwchl	calico-system
ca-certs	kube-system
kube-api-access-rfwbm	monitoring-system
containers	monitoring-system
kube-api-access-rrgbw	azimuth
kube-api-access-xdzxf	postgres-operator
kube-api-access-v4r69	monitoring-system
sys	monitoring-system
journal	monitoring-system
kube-api-access-dgkvx	tigera-operator
proc	monitoring-system
containers	monitoring-system
kube-api-access-86gtj	monitoring-system
containers	monitoring-system
machine-id	monitoring-system
kube-api-access-7bgcw	openstack-system
kube-api-access-w55wn	capi-addon-system
kube-api-access-g4z29	node-problem-detector
journal	monitoring-system
cert-volume	keycloak-system
lib-modules	calico-system
kube-api-access-2vfvt	openstack-system
tls-assets	monitoring-system
kube-api-access-mdwjv	openstack-system
kube-api-access-qj75d	keycloak-system
kube-api-access-pmcr9	capi-kubeadm-bootstrap-system
kube-api-access-7bgcw	openstack-system
kube-api-access-cml7x	node-problem-detector
usr-share-ca-certificates	kube-system
kube-api-access-ghwwn	cert-manager
kube-api-access-2vv9g	calico-system
kube-api-access-l5z5z	node-problem-detector
pgbackrest-server	keycloak-system

Table 9: List of Kubernetes containers running as root (3 of 4)

Name	Namespace
kube-api-access-lds6w	kubernetes-dashboard
k8s-certs	openstack-system
kube-api-access-gl6mb	monitoring-system
ca-certs	kube-system
kube-api-access-2rjqb	cert-manager
kube-api-access-mdwjv	openstack-system
sys	monitoring-system
k8s-certs	kube-system
kube-api-access-v4stc	kube-system
proc	monitoring-system
sys	monitoring-system
flexvolume-dir	openstack-system
kube-api-access-86gtj	monitoring-system
root	monitoring-system
etc-ca-certificates	kube-system
log	node-problem-detector
pods	monitoring-system
usr-share-ca-certificates	kube-system
localtime	node-problem-detector
log	node-problem-detector
root	monitoring-system
kube-api-access-2vfvt	openstack-system
localtime	node-problem-detector
kube-api-access-thtsc	monitoring-system
kube-api-access-2vfvt	openstack-system
kube-api-access-8bnkc	openstack-system
kube-api-access-7bgcw	openstack-system
kube-api-access-8bnkc	openstack-system
kube-api-access-8bnkc	openstack-system

Table 10: List of Kubernetes containers running as root (4 of 4)

Recommendations

Review if running as privileged containers is necessary for all containers or if the number can be reduced by using more restrictive configurations.

2.10 Kubernetes Containers with the Privilege Escalation Flag Set (Vulnerability - Low)

During our engagement, we identified several containers with the privilege escalation flag set.

In some cases, setting the privilege escalation flag is necessary to perform operations that require elevated permissions, such as accessing host devices, modifying kernel parameters, or accessing hardware such as GPUs. However, setting the privilege escalation flag is a security risk because it grants the container extensive access to the host system.

Name	Namespace
calico-node	calico-system
csi-node-driver-registrar	calico-system
calico-node	calico-system
calico-csi	calico-system
csi-node-driver-registrar	calico-system
calico-csi	calico-system
csi-node-driver-registrar	calico-system
calico-csi	calico-system
cinder-csi-plugin	openstack-system
calico-node	calico-system
cinder-csi-plugin	openstack-system
cinder-csi-plugin	openstack-system

Table 11: Containers with the Privilege Escalation Flag Set

Recommendations

It looks like all of the containers are doing storage operations and might need the access. Review if setting the privilege escalation flag is necessary for all of the containers, or if certain containers can be reduced to more specific privileges.

2.11 Kubernetes Containers Running as Root (Vulnerability - Low)

During our engagement, we identified several containers running as the root user.

In some cases, running containers as root is necessary to perform privileged operations, such as mounting volumes or changing network settings. However, running containers as root is a security risk because it grants the container full access to the host system.

Name	Namespace
operator	azimuth
sshd	azimuth
kube-controller-manager	kube-system
kube-proxy	kube-system
dex	az-stackhpc-dev
ui	azimuth
sync	azimuth
node-driver-registrar	openstack-system
sshd	azimuth
cluster-api-addon-provider	capi-addon-system
operator	postgres-operator
cert-manager-controller	cert-manager
calico-node	calico-system
etcd	kube-system
dex	az-stackhpc-lab
cert-manager-cainjector	cert-manager
operator	azimuth
cert-manager-webhook	cert-manager
coredns	kube-system
dex	az-stackhpc-blob
helm-dashboard	monitoring-system
keycloak	keycloak-system
liveness-probe	openstack-system
keycloak-operator	keycloak-system
operator	azimuth
pgbackrest-config	keycloak-system
kube-proxy	kube-system
tigera-operator	tigera-operator
statsd-exporter	azimuth
csi-node-driver-registrar	calico-system
calico-node	calico-system
pgbackrest-config	keycloak-system
operator	azimuth
calico-csi	calico-system
kube-apiserver	kube-system
csi-node-driver-registrar	calico-system
promtail	monitoring-system
sshd	azimuth
operator	capi-janitor-system
node-problem-detector	node-problem-detector
node-problem-detector	node-problem-detector
calico-csi	calico-system

Table 12: List of Kubernetes containers running as root (1 of 2)

Name	Namespace
database	keycloak-system
csi-resizer	openstack-system
node-driver-registrar	openstack-system
csi-node-driver-registrar	calico-system
pgbackrest	keycloak-system
node-driver-registrar	openstack-system
kube-scheduler	kube-system
liveness-probe	openstack-system
dex	az-stackhpc-azimuth
liveness-probe	openstack-system
calico-csi	calico-system
liveness-probe	openstack-system
replication-cert-copy	keycloak-system
coredns	kube-system
csi-attacher	openstack-system
dex	az-stackhpc-ci
csi-provisioner	openstack-system
api	azimuth
csi-snapshotter	openstack-system
registrar	azimuth
exporter	keycloak-system
cinder-csi-plugin	openstack-system
calico-node	calico-system
kube-proxy	kube-system
cinder-csi-plugin	openstack-system
pgbackrest	keycloak-system
promtail	monitoring-system
promtail	monitoring-system
node-problem-detector	node-problem-detector
cinder-csi-plugin	openstack-system
cinder-csi-plugin	openstack-system

Table 13: List of Kubernetes containers running as root (2 of 2)

Recommendations

Review if running containers as root is necessary or if it can be run with reduced privileges.

List of Figures

8figure.caption.3

List of Tables