zulusec

Security Audit Findings Report

Cloud and Kubernetes environment assessment

Prepared for Northwind Freight, Inc.

This is a sample report. Northwind Freight is a fictional company and the environment described here is synthetic, assembled from patterns we see repeatedly in real environments. No client data appears in this document. It is published to show the format, depth, and reasoning of a ZuluSec findings report before you commission one.

Engagement	**Security Audit, Standard**
Environments	**AWS production account, EKS cluster, corporate network**
Fieldwork	**Five business days**
Report version	**1.0, final**
Prepared by	**ZuluSec LLC, Birmingham, Alabama**
Classification	**Confidential to the client**

1. Executive summary

Northwind Freight asked ZuluSec to audit the security of its production AWS account, its Kubernetes cluster, and the corporate network that connects to them. The work was a read-only review of configuration, access, and architecture. We did not exploit anything and we did not change anything.

The environment is well built in most respects and badly exposed in three. The engineering fundamentals are here: infrastructure is largely in code, workloads are containerized, and the team patches quickly. What is missing is a boundary between the parts of the system that face the world and the parts that hold customer data, plus any practical ability to notice an intruder.

We found eight issues: three high, three medium, two low. The three high findings are not theoretical. Each one can be reached today, from the public internet, by someone with no credentials and no inside knowledge, and each leads to customer data.

3	3	2	8
HIGH	MEDIUM	LOW	TOTAL

If you only do three things

1. **Close the public storage bucket holding customer exports (ZS-01).** Roughly 240,000 customer records are downloadable by anyone who guesses the bucket name and a dated file name. This is the finding that would become a disclosure event. About two hours of work.
2. **Remove the anonymous administrative binding on the Kubernetes cluster (ZS-02).** An unauthenticated request from the internet can currently read every secret in the cluster, including the production database password. One command, then a credential rotation.
3. **Revoke the AWS access key published in a public git repository (ZS-03).** The key is still active and still has broad permissions. Deleting the commit did not remove it.

> **The theme underneath the findings.** Northwind treats the network as the security boundary: once a request is inside the VPC or inside the cluster, it is trusted. Three of the eight findings are variations on that single assumption. Fixing the individual issues closes today's exposure. Moving identity and authorization to the front of every request is what stops the next one, and it is the natural follow-on project.

What is working

- Infrastructure is defined in code and peer reviewed, so fixes here are durable rather than manual.
- Operating system and container base images are current, with no end-of-life systems in scope.
- Production databases are encrypted at rest and backups run and are retained.
- No shared administrative accounts. Every human has an individual identity.

2. Scope and method

What was assessed

ENVIRONMENT	DETAIL
AWS production account	Single account, two regions, 61 EC2 instances, 24 S3 buckets, 4 RDS instances
Kubernetes	One EKS cluster, 3 nodes, 38 workloads across 6 namespaces
Corporate network	One site, 22 systems, site-to-site VPN into the production VPC
Identity	Cloud identity provider, 34 user accounts, 9 with administrative roles

How we worked

All access was read-only, granted through a dedicated audit role scoped at the kickoff call and authorized in writing before fieldwork began. We reviewed configuration, identity and access policy, network reachability, logging coverage, and data exposure, then traced how a weakness in one area could be chained into another. Findings were validated by observation only.

Explicitly out of scope

- Exploitation of any finding. This was an audit, not a penetration test.
- Denial of service, load testing, and any destructive action.
- Social engineering, phishing, and physical access.
- Third-party SaaS the client does not own or control.
- Application source code review beyond secrets and configuration exposure.

How severity is assigned

SEVERITY	MEANING
HIGH	Reachable today and leads directly to customer data, credentials, or administrative control. Fix this week.
MEDIUM	Meaningfully raises the odds or the cost of a breach, or would prevent you from detecting one. Fix this month.
LOW	No active exposure today, but it turns a future small mistake into a large incident. Fix this quarter.

3. Findings at a glance

ID	SEVERITY	FINDING	EFFORT
ZS-01	HIGH	Customer data exportable from a public storage bucket	~2 hours
ZS-02	HIGH	Unauthenticated administrative access to the Kubernetes cluster	~1 day, plus secret rotation
ZS-03	HIGH	Active cloud access key published in git history	~2 hours
ZS-04	MEDIUM	A breach in progress would not be detected	~1 day
ZS-05	MEDIUM	Multi-factor authentication not enforced for administrators	~half a day
ZS-06	MEDIUM	Corporate network reaches production without restriction	~1 week
ZS-07	LOW	Network rules allow inbound access from anywhere	~half a day
ZS-08	LOW	Standing administrative privilege that is never exercised	~half a day

4. Detailed findings

ZS-01 HIGH

Customer data exportable from a public storage bucket

Affected: one S3 bucket in the production account, holding nightly customer exports

WHAT WE FOUND

A bucket used for nightly customer exports has Block Public Access off at both the account and the bucket level, and a bucket policy granting `s3:GetObject` to `"Principal": "*"`. The objects are readable by anyone on the internet with no AWS account and no credentials. The grant covers reading an object, not listing the bucket, so an attacker needs the object key as well as the bucket name. Exports are named `customers-YYYY-MM-DD.csv`, which makes the key as guessable as the name. The most recent export is 118 MB of CSV; the nightly job that writes it emits one row per customer, approximately 240,000 of them, including names, email addresses, physical addresses, and shipment history.

Server access logging is not enabled on the bucket, so there is no record of who has read from it.

EVIDENCE

```
# anonymous listing is refused: the policy grants read, not enumeration
$ aws s3 ls s3://REDACTED-exports --no-sign-request
An error occurred (AccessDenied) when calling the ListObjectsV2 operation: Access Denied

# the object itself answers a plain unauthenticated HTTP request: no SDK,
# no AWS account, no credentials, no request signature. Headers trimmed.
$ curl -sI https://REDACTED-exports.s3.amazonaws.com/customers-2026-08-11.csv
HTTP/1.1 200 OK
Last-Modified: Tue, 11 Aug 2026 02:00:14 GMT
Content-Type: text/csv
Content-Length: 118237442
Server: AmazonS3

# first 4 KB only. We did not download the export.
$ curl -s -r 0-4095 https://REDACTED-exports.s3.amazonaws.com/customers-2026-08-11.csv | head
-2
id,name,email,address,last_shipment
1041,REDACTED,REDACTED,REDACTED,2026-08-09
```

WHY IT MATTERS

Bucket names are not secrets. They appear in URLs, front-end code, DNS records, and third-party scanning datasets, and internet-wide scanning for readable buckets is continuous and automated. A dated filename is not a secret either, so the missing list permission buys you nothing. Because access logging is off, you cannot demonstrate that the data was not taken, which in most breach-notification regimes means you must assume that it was.

RECOMMENDED FIX

1. Enable Block Public Access at the account level. This overrides any bucket policy, including ones written in future by mistake.
2. Remove the wildcard grant from the bucket policy.
3. Enable server access logging, or CloudTrail data events, on all buckets holding customer data.
4. If a bucket genuinely must serve public files, keep the bucket private and put a CDN in front of it.
5. Treat the exposed export as compromised and follow your incident and notification process.

Effort: about 2 hours, plus incident handling

Maps to: CIS AWS Foundations Benchmark, S3 block public access · NIST SP 800-53 Rev 5 AC-3, AC-6, AU-2

ZS-02 HIGH

Unauthenticated administrative access to the Kubernetes cluster

Affected: production EKS cluster, all six namespaces

WHAT WE FOUND

A `ClusterRoleBinding` named `dashboard-access` binds the `cluster-admin` role to the `system:anonymous` user. Kubernetes assigns that identity to any request arriving without valid credentials. The cluster's API server endpoint is also reachable from the public internet rather than restricted to known networks.

Together these mean an unauthenticated request from anywhere on the internet holds full administrative control of the cluster. Commit history indicates the binding was added during a dashboard troubleshooting session and was not removed afterward.

EVIDENCE

```
# the binding, as read from the cluster. Metadata trimmed.
$ kubectl get clusterrolebinding dashboard-access -o yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: dashboard-access
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: system:anonymous

# what that permits. curl rather than kubectl on purpose: kubectl would read
# our kubeconfig and present our own client certificate or exec credential,
# which would prove nothing. eks-ca.pem authenticates the server to us, not us
# to it, so this request carries no credential and the API server resolves it
# to system:anonymous.
$ curl -s --cacert eks-ca.pem https://REDACTED.eks.amazonaws.com/api/v1/secrets \
  | jq -r '.items[] | "\(.metadata.namespace)  \(.metadata.name)"'
prod  db-credentials
prod  stripe-api-key
# 34 further secrets omitted from this report
```

WHY IT MATTERS

Cluster administrator is the end state, not the starting point. From it, an attacker reads the production database credentials shown above, schedules a workload of their choosing inside your network, and uses the node's instance role to move into the wider AWS account. No password is guessed and no software vulnerability is required.

RECOMMENDED FIX

1. Delete the binding: `kubectl delete clusterrolebinding dashboard-access`.
2. Rotate every secret in the cluster. Assume all of them are known.
3. Restrict the API server endpoint to authorized networks, or make it private.
4. Audit remaining bindings for `system:anonymous`, `system:unauthenticated`, and wildcard subjects.
5. Add a policy check in CI that fails any manifest binding a privileged role to an anonymous subject.

Effort: minutes to close the exposure, about a day for the binding audit and the CI check, plus rotation of 36 cluster secrets across six namespaces

Maps to: CIS Kubernetes Benchmark, RBAC and service account recommendations (restricting cluster-admin, minimizing access to secrets) and the control plane recommendation against anonymous authentication · NIST SP 800-53 Rev 5 AC-2, AC-6

ZS-03 HIGH

Active cloud access key published in git history

Affected: one IAM user, one public repository

WHAT WE FOUND

A long-lived AWS access key for an IAM user named `deploy-bot` was committed to a public repository in a `.env` file. The file was deleted in a later commit, but the key remains readable in history, and the key is still active. The user holds broad permissions including read access to the storage layer.

The key was first committed 14 months ago. It has never been rotated.

EVIDENCE

```
# full history scan of the public repository. --redact=80 leaves the first
# four characters of the secret in gitleaks' own output; the key id is masked
# the same way everywhere it appears below.
$ gitleaks git --no-banner --no-color --redact=80 .
Finding:     AKIA...
Secret:      AKIA...
RuleID:      aws-access-token
Entropy:     3.684184
File:        .env
Line:        3
Commit:      3f9a1c7e2b5d8a1049c3f6b7e0d2a915c8146b3f
Author:      REDACTED
Email:       REDACTED
Date:        2025-06-14T09:41:07Z
Fingerprint: 3f9a1c7e2b5d8a1049c3f6b7e0d2a915c8146b3f:.env:aws-access-token:3
# .env was deleted in 8b2e440. The blob is still reachable in history.

# key state, read through the audit role. We did not authenticate with the
# key itself: an audit does not use a credential it has just found.
$ aws iam list-access-keys --user-name deploy-bot
{
    "AccessKeyMetadata": [
        {
            "UserName": "deploy-bot",
            "Status": "Active",
            "CreateDate": "2025-06-14T09:42:11+00:00",
            "AccessKeyId": "AKIA..."
        }
    ]
}
```

WHY IT MATTERS

Removing a secret from the current checkout does not remove it from history, and public repositories are scraped continuously for credentials. A key exposed for fourteen months should be treated as known to others. Because it is a long-lived key rather than a temporary role credential, it works from anywhere with no additional checks.

RECOMMENDED FIX

1. Deactivate and delete the key now. Do not wait for the cleanup in step 3.
2. Review CloudTrail for use of this key from unfamiliar addresses over the retained period.
3. Purge the secret from history, then treat it as burned regardless.
4. Replace the long-lived key with a short-lived role credential for CI.
5. Add automated secret scanning to pre-commit hooks and to CI so the next one is caught before it merges.

Effort: about 2 hours, plus log review

Maps to: CIS AWS Foundations Benchmark, IAM access key rotation · NIST SP 800-53 Rev 5 IA-5

ZS-04 MEDIUM

A breach in progress would not be detected

Affected: both regions, all accounts in scope

WHAT WE FOUND

Audit logging is enabled in one region only, so activity in the second region is not recorded at all. Managed threat detection is switched off. Where logs do exist, no alert is routed to a person or a channel, and audit log retention is 90 days.

WHY IT MATTERS

Every finding in this report is more damaging if nobody notices it being used. An attacker working in the unlogged region leaves no trace, and even in the logged region the record exists only for someone who thinks to go looking. Detection is also what turns an incident from an open-ended question into a bounded one: without it you cannot answer what was taken, which forces you to assume the worst in any disclosure.

Ninety days of retention is the same problem one step later. Discovery is not the start of the intrusion, and the questions that follow it, when they first got in, what they touched, which credentials they used, are answered from logs written well before anyone thought to look. ZS-03 is the concrete case in this environment: that key has been exposed for fourteen months and your retained window covers the last three of them. Where a compliance regime applies to you, its audit retention requirement will also run longer than ninety days.

RECOMMENDED FIX

1. Enable multi-region audit logging to a dedicated, access-restricted bucket with log file validation on.
2. Turn on managed threat detection in every region you operate in.
3. Route findings to somewhere a human actually reads, and agree who owns triage.
4. Extend retention to at least one year for audit logs.
5. Start with a small number of high-signal alerts: root account use, IAM policy changes, public storage grants, disabling of logging.

Effort: about 1 day

Maps to: CIS AWS Foundations Benchmark, multi-region CloudTrail logging and the monitoring and alarm recommendations · NIST SP 800-53 Rev 5 AU-6, SI-4

ZS-05 MEDIUM

Multi-factor authentication not enforced for administrators

Affected: 9 accounts holding administrative roles, of 34 total

WHAT WE FOUND

Multi-factor authentication is available and used by most of the engineering team, but it is not required by policy. Four of the nine accounts with administrative roles can authenticate with a password alone, and the cloud root account has no hardware factor registered.

WHY IT MATTERS

Password-only administrative access is the single most reliable path into a cloud environment, because it does not require any technical vulnerability. Credential reuse and phishing are routine, and voluntary adoption predictably leaves exactly the accounts that matter most uncovered.

RECOMMENDED FIX

1. Require phishing-resistant multi-factor authentication for every account with administrative access, enforced rather than requested: a service control policy or an IAM policy condition on `aws:MultiFactorAuthPresent` for AWS access, and the equivalent sign-in enforcement rule at your identity provider for federated access.
2. Register a hardware factor on the root account and store it physically. Use the root account only for tasks that require it.
3. Extend the requirement to all users on a scheduled date, with the exception list reviewed and empty.

Effort: about half a day, plus rollout

Maps to: CIS AWS Foundations Benchmark, MFA for the root user account (including hardware MFA) and MFA for all IAM users with a console password · NIST SP 800-53 Rev 5 IA-2

ZS-06 MEDIUM

Corporate network reaches production without restriction

Affected: site-to-site VPN, 22 corporate systems, production VPC

WHAT WE FOUND

The corporate site connects to the production VPC over a site-to-site VPN with no filtering between them. Any device on the office network, including laptops, printers, and guest devices on the same segment, can reach production databases and internal services directly on their service ports.

WHY IT MATTERS

This makes every corporate endpoint a production endpoint. One phished laptop or one unmanaged device is enough to reach the data tier, without any further compromise. It is also the finding that most amplifies the

others: it means an intrusion anywhere in the office becomes an intrusion in production.

RECOMMENDED FIX

1. Filter the VPN so only named source systems reach named production services on required ports. Deny by default.
2. Separate guest and unmanaged devices onto their own segment with no route to production.
3. Move administrative access to production behind an identity-aware access layer, so reaching it depends on who and what is asking rather than where the packet came from.

Effort: about 1 week

Maps to: CIS Critical Security Controls, secure network architecture and traffic filtering between network segments · NIST SP 800-53 Rev 5 SC-7, AC-4

ZS-07 LOW

Network rules allow inbound access from anywhere

Affected: 14 security groups

WHAT WE FOUND

Fourteen security groups permit inbound traffic from `0.0.0.0/0` on administrative and database ports. On the instances currently attached to these groups, the corresponding services are not listening, so there is no active exposure today. That is a fact about what happened to be running during the fieldwork window, not a fact about the rules, and nothing keeps it true.

WHY IT MATTERS

These are loaded guns rather than fired ones. The rules are permanent while the safety is incidental: the day a service starts listening on one of those ports, or an instance is attached to one of these groups by an autoscaling policy, the exposure becomes real with no change to the rule and no review.

RECOMMENDED FIX

1. Scope each rule to the specific source ranges that need it.
2. Remove groups that are unused.
3. Add a check in the infrastructure pipeline that rejects world-open administrative and database ports.

Effort: about half a day

Maps to: CIS AWS Foundations Benchmark, unrestricted ingress to remote server administration ports · NIST SP 800-53 Rev 5 SC-7

ZS-08 LOW

Standing administrative privilege that is never exercised

Affected: 3 IAM roles, 2 service accounts

WHAT WE FOUND

Three roles hold full administrative permissions and, according to access analyzer data over the last 90 days, have used none of them. Two service accounts hold write permissions to storage they only ever read from.

WHY IT MATTERS

Unused permission is pure downside. It does nothing for you day to day and everything for an attacker who acquires the identity. Right-sizing these does not change how the system works but meaningfully reduces what a single compromised credential is worth.

RECOMMENDED FIX

1. Right-size each role to the permissions actually used over the observed period.
2. Change the two service accounts to read-only.
3. Schedule a quarterly access review, and prefer temporary elevation over standing administrative rights.

Effort: about half a day

Maps to: CIS AWS Foundations Benchmark, avoiding IAM policies that grant full administrative privileges · NIST SP 800-53 Rev 5 AC-6

5. Remediation plan

Ordered by risk reduction per hour of effort, not by severity alone. The first block is under two working days of configuration work, plus the credential rotations it triggers, and it removes every path that is currently reachable from the internet.

WHEN	FINDINGS	OUTCOME
This week	ZS-01, ZS-02, ZS-03	No unauthenticated path from the internet to customer data. Includes rotating the cluster secrets and the exposed key, which is the part teams most often defer and should not.
This month	ZS-04, ZS-05	You would see an intrusion, and administrative accounts stop being password-only. Together these convert an unbounded incident into a bounded one.
This quarter	ZS-06, ZS-07, ZS-08	A compromised laptop or credential no longer reaches production by default. This is where the architectural work sits.

Then the structural piece

Findings ZS-02, ZS-06, and ZS-07 all follow from treating network position as proof of trust. Once the immediate items are closed, the highest-value next step is moving authorization to the front of every request: strong identity everywhere, access decided per request on who and what is asking, and segmentation that assumes a breach has already happened. That is a project rather than a fix, and it is the difference between closing these eight findings and not generating the next eight.

> **What we would want to see at a re-check.** The three high findings closed and their credentials rotated, logging and detection producing alerts a human has acknowledged, and administrative multi-factor enforced by policy rather than by convention. That is a materially different risk position from where the environment stands today.

6. Limitations

This report reflects the scope agreed at kickoff, the access provided, and the state of the environment during the fieldwork window. It is a point-in-time review.

- An audit does not prove the absence of vulnerabilities. It reports what was found in the agreed scope, in the time agreed.
- No finding was exploited. Impact is assessed from configuration and reachability, not from a demonstrated compromise.
- Systems outside the listed scope were not examined, including third-party services the client does not control.
- Configuration changes made after the fieldwork window are not reflected here.
- Where we note that data should be assumed compromised, that is a risk judgment, not a confirmed breach determination.

Review coverage

Every domain below was reviewed in each in-scope environment.

DOMAIN	COVERED
Identity and access	Accounts, roles, permission boundaries, multi-factor coverage, standing privilege, key age and rotation
Network exposure	Internet-reachable services, security group and firewall rules, segmentation, remote access paths
Data exposure	Storage permissions, public access configuration, encryption at rest, backup exposure, secrets in code
Kubernetes	RBAC, control plane exposure, secrets handling, network policy, workload privilege
Logging and detection	Audit log coverage and retention, threat detection, alert routing and ownership