2021-12-07 Registry Advisory

Summary

Registry v3.3.0 introduced CO-specific API users, which can be either privileged (having full access to the CO via the API) or unprivileged (having no specific access unless granted). Certain foreign keys were not properly validated within some APIs, resulting in the potential for data leakage.

Severity

The severity of this issue is *medium*, as a privileged API user is required to leak data.

Exposure

The exposure will generally be *low*, as this advisory only meaningfully affects multi-tenant deployments, and only those that have enabled CO-specific API users.

Recommended Mitigation

Deployments not using the described configuration need not take any action, though should plan an upgrade as soon as plausible in case CO-specific API users are created later.

Deployments using the described configuration should immediately upgrade to Registry v4.0.1, or to develop commit 13a0c8cfdd or later.

Deployments may also perform an audit, as described in *Discussion*, below.

Alternate Mitigations

Deployments may alternately disable any privileged CO-specific API users until an upgrade can be performed.

Discussion

Registry v3.3.0 introduced CO-specific API users, which can be either privileged (having full access to the CO via the API) or unprivileged (having no specific access unless granted). Previously, the REST API was only available to platform-wide superusers.

As part of the patch for the 2021-05-24a Registry Advisory, improved validation routes were introduced. These changes applied to edit operations, not add operations, as add operations generally have other checks in place that prevented the originally described issue. Certain APIs did not implement proper add validation.

Unlike the original May advisory, It does not appear possible to escalate privileges. It is possible, however, to leak some data across COs in a read-only manner.

To check for exploits, SQL queries can be used to compare the `actor_identifier` to the CO of the relevant record. For example:

```
WITH t1 AS (
  SELECT cm_identifiers.id,cm_identifiers.actor_identifier,cm_org_identities.co_id
  FROM cm_identifiers
  INNER JOIN cm_org_identities on cm_identifiers.org_identity_id=cm_org_identities.id
  WHERE cm_identifiers.actor_identifier IN (
    SELECT username
    FROM cm_api_users
    WHERE co_id > 1
    AND privileged = true
  )
), t2 AS (
  SELECT username,co_id
  FROM cm_api_users
  WHERE co_id > 1
  AND privileged = true
)
SELECT * FROM t1
LEFT JOIN t2 on t1.actor_identifier=t2.username
WHERE t1.co_id <> t2.co_id;
```

Tables to examine include `cm_co_person_roles`, `co_person_roles`, `co_departments`, `co_email_lists`, `co_groups`, `co_org_identity_links`, `co_petitions`, `co_services`, `co_terms_and_conditions`, `history_records`, and `co_unix_cluster_groups`.

References

- CO-2146
- CO-2294