2021-04-20 - US-1 Full Service Disruption

Updated 2021-07-01

Detailed Root Cause Analysis (RCA)

We understand the frustration that this service disruption caused to our US-1 customers, and we are deeply sorry that this occurred. We know you rely on Auth0 to provide a critical service, and during this incident, our service failed to deliver. We apologize.

Summary

On April 20th, 2021, our US-1 Production and Preview environments experienced a full service disruption. During this time customers were unable to complete authentication flows. The requests that passed through had extremely high latency (up to 100x on average). This service disruption was due purely to performance degradation on our internal platform systems, was not the result of any third-party service, and did not result in any data leakage. After investigation, we are confident that the disruption was not caused by, and did not result in, a security compromise.

Details on a per-service impact are below—all times in UTC.

Service	Start	End
US-1 Production	15:27	19:47
US-1 Preview	15:27	20:04
All Regions - Degraded Management Dashboard	15:27	18:18
All Regions - Management Dashboard	18:18	20:04
Support Center	15:27	19:47
status.auth0.com	15:48	16:50
US-1 User Import/Export	15:27	23:25
US-1 User Search	15:27	23:25

From our analysis, we determined that the incident was caused by the exhaustion of resources for our feature flag service and was further exacerbated by three poorly performing queries. This service provides feature flag and configuration data to our front-end API nodes and is described in greater detail later in this RCA. The feature flag service caused its clients to fail back to the server rather than the cache. It then began querying the database directly with every request, overwhelming our

primary operational database, and leading to increased query response times. The increased query response times caused a cascading failure as calling services repeatedly retried their API calls after reaching timeout thresholds.

We responded immediately and attempted to solve the issue in several ways, initially focusing on the query that we believed to have caused the problem. We executed a controlled step-down to failover the primary database to one of the replica databases as a part of that effort. This replica database resides in a different availability zone. (We considered executing a failover of the entire US-1 environment to another region, but opted not to do so because, since the disruption was due to load and not unexpected infrastructure failure, failover of the entire environment would not have resolved this type of performance issue.) When the query improvements and failover did not improve system performance we shifted to reducing the load on the system to allow the database to recover.

We took a variety of steps to reduce load, which are detailed later on in this RCA. At 19:25, we scaled down the maximum number of front-end API instances, which was the critical step in reducing load sufficiently to allow the system to recover. We began to see authentication flows completing successfully at 19:36, with core authentication services fully recovered at 19:47.

What Happened

The components that set off the chain of events in this incident are the feature flag service and the overloaded database. The feature flag service provides our front-end API nodes with the configuration values for tenants. This service includes a caching layer to reduce database load. A client on each front-end API instance queries this service. If the cache does not respond within the timeout, the client falls back to querying for the data directly from the feature flag service. The service then queries the primary database.

With regards to the overloaded database, we identified three poorly performing queries which impacted performance. These queries are used during normal system operation and have been in use for months without issue. They are not ad-hoc or one-off queries. It was not until the queries were combined with other heavy loads that their poor performance became problematic. Details on the problems with these queries are below.

1. The first poorly performing query had a primary collection that did not have an index, and this caused an unbounded scan of a very large collection.
2. The second query ran a collection scan that examined a high number of documents.
3. The third query was rare in frequency, but very resource-intensive. Its cleanup cascaded through many collections.

Our analysis has concluded that there were three causes that - when compounded simultaneously - resulted in this service disruption. In our summary RCA we noted initially suspecting that slow-performing queries were the trigger, but our investigation has concluded that the initial triggering factor for this incident originated from the feature flag service. An increase in traffic exceeded the caching capacity of that service and caused it to stop responding in a timely manner. This, in turn, caused the client on each front-end API node to fall back to querying the service directly

instead of the cache. The feature flag service then queried the primary database directly. The additional load from these direct queries, combined with the three poorly performing queries mentioned above, resulted in the database becoming overloaded and exceeding its available disk I/O.

As front-end API nodes stopped receiving timely responses, they retried performing the original queries, and more incoming requests were added to the queue. This increase in load caused the front-end API nodes to scale out from 37 to 100 nodes, further loading the database and preventing the system from recovering.

The system is designed to bypass our feature flag service cache should it become unresponsive. Under normal circumstances, latency increases as the database handles this additional load but the service returns to normal after the traffic spike subsides. In this incident, the database was also overloaded and unable to respond in a timely manner. Our monitoring did detect that the feature flag service had fallen back to the database, but in the moments during the incident, this was mistakenly attributed to a problem in the front end API services.

Exacerbating the above, the Auth0 system is also designed to scale up our front-end API nodes when their resource utilization rises beyond a threshold. The increased number of in-flight requests due to long query times caused increased utilization, causing them to quickly scale to the maximum number of instances. More front end API nodes connecting directly to the feature flag service shifted the load from the feature flag service cache to the already poorly performing database, causing a set of cascading failures.

How We Responded

At 15:27 UTC on April 20th, 2021, many different alerts were triggered for our US-1 Production environment for login failures, latency, memory usage, errors, and database connections. We declared an incident at 15:30, and the incident response team began investigating. Our initial discovery showed poor performance across various services. Query response times had increased significantly, and internal SLOs dropped. Our primary database was not reporting metrics, and the front-end API servers had scaled up significantly.

At 15:45, we posted the incident to our Status Page as a major outage and described it as “An increased error rate.” At that time, the scale of the impact was not yet clear, and our reporting metrics were still showing a level of success. Shortly after, at 15:48, alerts triggered and paged the team responsible for the status page. This spawned a separate incident, which is detailed in its own section below.

One of our first steps was to disable promotion and revert to a previous release. We did so at 16:00 and 16:05, rolling back to the previous week’s versions. At 16:04, we reduced the maximums and targets for our auto-scaling groups. At approximately 16:10, our primary database began reporting metrics again, increasing our visibility into what was happening. A portion of the incident response team then started investigating concerning queries.

Our investigation continued on multiple fronts, including into our feature flag service, database performance, and auto-scaling group size. Representatives from every engineering team joined the incident and worked quickly to determine how their respective services were performing, rule out other causes, and disable non-critical services. This was an all hands on deck effort, with hundreds of people from across the organization contributing to the investigation and providing what help they could offer, while taking care not to disrupt the efforts of the core incident team.

At 17:08, the team deployed a change to add an index designed to help with one of the most concerning queries. The index builds were completed at 17:21 but did not have any positive effect. Our later investigation would show that the added index did not address the query performance (but we note that this incorrect index was corrected shortly after the incident was resolved). When we did not see an immediate impact, our primary focus shifted from targeted changes to considering all options to reduce or reset load so the system could recover.

From 17:55 until 19:43, we took a variety of steps to reduce load as follows:

- 17:34 - Performed a controlled step-down of the primary database and promoted one of the replica databases to primary
- 17:55-19:25 - Adjusted auto-scaling groups to cause front end API nodes to recycle
- 18:31 - Turned off feature flag service
- 19:09 - Turned off user exporting features
- 19:25 - Scaled down front-end API nodes from 100 to 45
- 19:39 - Scaled up secondary database node
- 19:43 - Turned off user importing features
- 19:43 - Implemented a reduced rate limit on our Authentication API

The 19:25 scale down from 100 to 45 front-end API nodes was the critical step that sufficiently reduced load and allowed the system to recover. We received reports of slow but successful logins at 19:36, with the restoration of core authentication services at 19:47.

Following the return of service, we began slowly restoring non-critical services and increasing the primary database IOPS. All services were returned to normal functionality by 23:25.

Status Page Failure

We revamped our Status Page in October 2020 to provide more clarity on the status and uptime of our various environments. During this incident, the new Status Page became overwhelmed by traffic and was not responding from 15:48 to 16:50.

A separate team spun off to focus on restoring the Status Page. They initially believed this was a capacity issue with the servers hosting it, but scaling those servers up had no impact. Further investigation revealed that we were calling our Status Page provider's API each time it was loaded and, with the increased amount of traffic, these calls were being rate-limited. While some customers

were still able to access the site during the periods we were not being rate-limited, the majority of users saw either an error page or stale data. Upon realizing this, we pointed our DNS to the provider and fell back on their standard Status Page.

At 16:20, we posted an update to our Status Page about "Reports of our status.auth0.com being inaccessible for some customers," and advised to check the @auth0status Twitter account. We are now working on updates to the Status Page to cache results and avoid this rate-limiting in the future.

Mitigation Actions

We have taken this incident and our investigation very seriously, and have used all available resources to fully identify all compounding circumstances that led to this service disruption. As noted below, we have undertaken and completed many steps to rectify issues, and have established a timeline for additional action.

What We're Doing

Even though this was a rare confluence of factors, we immediately took action to increase capacity, improve query performance, and other steps to prevent a similar disruption from occurring in the future. In addition, we are taking the following actions as a result of this incident.

Update May 6, 2021: To provide transparency on the status of the improvements and corrections we have completed, we will be providing regular updates every two weeks in this document on work that has been completed.

Update May 20, 2021: Two action items were completed on May 19th and 20th. Our next update is targeted for June 3rd, 2021.

Update June 3rd, 2021: No new completions as of this report. All actions in progress with June targets are on track. Customer communication process updates are in progress, target moved out to June 2021. Our next update is targeted for June 17, 2021.

Update June 17th, 2021: One new action completion related to shifting load to our secondary databases. Our next update is targeted for July 1st, 2021.

Update July 1st, 2021: We have completed the remaining action items detailed below. This will be our final update to this RCA. While these action items are complete, resilience continues to be a major area of focus for our teams.

Systems/Platform

- Database
    - We have put in place indexes and query optimizations for poorly performing queries. (Completed before April 29th)

    - We have worked with our database software licensor to analyze the event and obtain their input on performance improvement opportunities. (Completed before April 29th)
    - We are accelerating an already planned effort to improve our database performance and resiliency. (Target June 2021, **Completed June 25th 2021**)
    - We are reviewing and optimizing indexes across a variety of queries, adding or removing as necessary. (Target June 2021, **Completed June 30th 2021**)
    - We are adjusting various services to prioritize database performance or shift load to the secondary database where possible. (Target June 2021, Completed June 9th 2021)
- System Performance
    - We are creating a playbook for how to most effectively reduce load during exceptionally high load service disruptions like this should a similar incident occur in the future. (Target May 2021, Completed May 19th 2021)
    - We are shifting resource intensive but not critical operations to services that can deal with those operations in a way that does not impact database performance. (Target June 2021, **Completed June 29th 2021**)
    - We are creating better error pages for the rare situations in which the system is unavailable, which will indicate there is a service disruption and where to check the status and get support. (Target June 2021, **Completed June 24th 2021**)
- Feature flag service
    - We have created a playbook for disabling the service should it be needed in a load shedding situation in the future. (Completed before April 29th)
    - We scaled up the feature flag cache to ensure it does not become overwhelmed during high traffic events. (Completed before April 29th)
    - We are reviewing and updating the feature flag caching instance configuration to improve monitoring and performance. (Target May 2021, Completed May 5th 2021)
- Other Areas
    - We are improving our Status Page to be more resilient to high traffic by implementing better caching mechanisms and other improvements. (Target May 2021, Completed April 26th)
    - We are fixing problems in our offline ticket submission in the Support Center, which did not work as expected during this incident. (Target June 2021, Completed: May 20 2021)

Customer Communications

- We are revisiting our customer communications processes for events such as these to provide more clarity and transparency as to when customers can expect next and future updates regarding incident status. (Target June 2021, **Completed June 30th 2021**)

General Incident Response

- We are reviewing and updating our incident response procedures to emphasize the importance of understanding the full impact of an outage across our customer base as early

in the incident lifecycle as possible. In this case, we knew it was impactful, but our processes did not put enough focus on understanding the depth of the impact, which led to missteps in communication to customers. (Target June 2021, **Completed June 29th 2021**)

- Since the incident we have made several adjustments to our internal Incident Management process with the goal of being more consistent and efficient in our response and communications. Most notably, we have:
  - Implemented additional tooling, dashboard, and alerting to monitor Auth0 availability % and latency SLOs
  - Improved Incident Commander training and established comprehensive resolution playbooks