The Post-Incident Review

Issue 2: March 2020

The Post-Incident Review

Issue 2: March 2020

Issue 3 arrives April 27, 2020

Digital and printable versions live at zine.incidentlabs.io, including instructions for assembling the zine. You can also sign up to get us directly in your inbox.

Got comments, suggestions, or compliments?
Drop us a line at zine@incidentlabs.io!

Editors's Note

What a difference two seasons can make.

When we launched Post-Incident Review, we excitedly brought stacks of hand-crafted printed copies to SREcon EMEA, looking forward to sharing how different post-incident reports felt when laid-out and printed like a journal. Mid-conference, people were coming up to us asking for a copy. We happily obliged.

Now, it looks like it'll be at least a few months before we'll get to see friends from far and wide. We were ready to print copies of Issue 2, but we realized things have changed—so we too had to think about how to adapt PIR. With everyone—us, included—at home, we thought, how can we keep the spirit of what we want to achieve going? It hit us: "What about leaning into our zine vibe?

What you'll find is a zine that now can be printed on just a handful of letter-size papers. There's an illustration by Denise Yu that's in black-and-white, just ready to be coloured in to give you (or maybe a little one) a break in the day. And, even better, you'll get more PIR, as we move to a monthly cadence.

It's difficult to imagine things returning to the way as they were, but part of the silver lining that comes from being students of the incident response process is that we can learn, and we can adapt, and we can try to find ways to make things better.

- Emil Stolarsky and Jaime Woo

"Server Outages and Increased API Errors"

Discord
Published on March 20, 2020

Summary
Discord was unavailable for most users for a period of an hour. The root cause is well understood and fixed. The bug was in our service discovery system, which is used by services within our infrastructure to discover one another. In this instance, service discovery is used by our real time chat services services in order to discover the RPC endpoint that they use to load data from our databases when you connect to Discord, or when a Discord server (or "guild") is created for the first time, or needs to be re-loaded from the database.

Timeline (all times are PDT)
[14:18] A set of nodes that serve our API traffic scales in to deal with growing load on the API cluster. This is an event that happens throughout the day. A single node, called `api-prd-main-m1ds` scaled in, and had an unexpected error when announcing to Service Discovery.

[14:19] Most of our Elixir services, which are used to handle our real time connections and chat message processing started to crash, resulting in an instantaneous loss of capacity, and causing a cascading failure into other dependent systems.

[14:21] Multiple alarms internally go off that signal a drop in key metrics we look at, as well as anomaly alerts for cluster utilization. A **SEV1** incident is declared. A phone h

bridge is set up, and all available engineers hop on to start investigating and establish internal and external communications.

[14:24] A status page incident is opened to let our users know that we're investigating: https://status.discordapp.com/incidents/62gt9cgjwdgf

[14:31] A tweet is posted, letting users know that we're looking into the issue, and to check the status page for more updates: https://twitter.com/discordapp/status/1239665509596983296

[14:23 to 14:43] A few engineers investigate why exactly we lost so much capacity on our real time systems, while other engineers focus on recovering service, restarting the lost capacity, and begin to throttle connections to Discord in order to help with service recovery. Additionally, we begin to stop database maintenance operations ("*anti-entropy repairs*") on two of our ScyllaDB clusters that would lead to resource starvation while everyone is attempting to re-connect.

[14:55] Engineers pinpoint the issue to be strongly correlated to a spike in errors in originating from our service discovery modules. It is determined that the service discovery processes of our API service had gotten into a crash loop due to an unexpected deserialization error. This triggered an event called "max restart intensity" where, the process's supervisor determined it was crashing too frequently, and decided to trigger a full restart of the node. This event occurred instantaneously across approximately 50% of the nodes that were watching for API nodes, across multiple

clusters. We believed it to be related to us hitting a cap in the number of watchers in etcd (the key-value store we use for service discovery.) We attempt to increase this using runtime configuration. Engineers continue to remediate any failed nodes, and restore service to users.

[15:07 to 15:26] The connection throttle is continually increased, allowing more users to reconnect as services recover.

[15:32] The status page incident is resolved, and service is deemed to be fully operational again.

[23:00] A mitigation for this issue is deployed to production in order to prevent this issue from recurring - once the root cause was fully understood.

Investigation and Analysis

The root cause of this outage was determined to be an invalid service entry being inserted into service discovery, causing a parse error while trying to deserialize that entry when loading it from etcd. Engineers worked to re-create this issue in a test environment, and were able to reproduce the same issue that was observed in production in our development environment.

- Discord uses an open source distributed, reliable, key-value store called etcd (https://github.com/etcd-io/etcd) in order to store service discovery information. Services that are discoverable announce themselves by setting a key in a specific directory in etcd that pertains to the cluster they are a part of. That key has a 60 second

TTL, and the service is responsible for heart-beating to etcd, to "re-announce" the key. Discord is using the etcd v2 API.

- At **14:18**, a node joined our API cluster, after being introduced by the Google Cloud managed instance group autoscaler. This is a normal event that happens hundreds of times a day, as utilization of the platform increases as we approach peak hours. This node logged an error while attempting to initially announce itself to etcd: `"http.client.RemoteDisconnected: Remote end closed connection without response"` Nearly immediately, almost all of our Elixir nodes logged that the service watcher for the "discord_api" service had crashed while attempting to parse the JSON metadata that should be stored in the key's value on etcd. These processes crash-looped briefly due to invalid JSON data in the etcd cluster, which lasted until that API node retried announcing itself to service discovery, fixing the "corrupt" key that had been written to etcd.

- Nodes announce themselves to etcd by issuing an HTTP PUT request that contains a urlencoded form body that contains the "value" of the key. In our case, this value is JSON encoded metadata that has information relevant to the discovery of the service. Our etcd client uses Python's built in HTTP client, and sends a PUT request (along with the Content-Length header) in one packet, and the request body in another packet. We determined that the connection was reset after sending the first packet, but *before* the second packet could be sent.

- A well-behaved HTTP server would see that it received a request specifying a Content-Length, and an incomplete or non-existent body, and reject this request. etcd is written in the Go programming language, and uses the Go standard library net/http HTTP request handlers for their v2 keys API. In order to parse the form body sent from clients, it uses the net.http/Request.ParseForm()method. This method does not check to see if the request body's length matches the length that was specified in the Content-Length header.

- This caused the key to be written with an empty string as the value, as the announce request was able to successfully send the headers, but did not send the body. This in turn caused an invalid key to be written to service discovery, which caused the downstream services to crash until the key was re-written when the announcing node retried.

Action Items and Response

Code within our service discovery system was not resilient to this type of failure - as it was not within our expectations that a key could be announced without a value due to a transient network error. Our service discovery system is resilient to various failure modes within etcd, however, we did not anticipate a client being able to write a corrupt record due to improper HTTP handling. golang's net/http module — which is in violation of the HTTP 1.1 specification in this specific case — introduced an additional failure case which we did not anticipate. It is expected that a Content-Length header and an incomplete body should be rejected by a

server as being incomplete. Unfortunately net/http does not check that the bytes read from the body portion of the request match the Content-Length header. We've since hardened our system to reduce the likelihood of this occurring and also handle invalid services being announced without crash looping.

- In order to reduce the likelihood of invalid keys being written to service discovery, we've modified our etcd clients to send their announce requests in a single TCP packet, instead of two. This means that the headers and body should either be received completely, or not at all.

- We've added additional error handling to ignore services that have a "corrupt" key value, just in case this issue does recur. The worst that will happen now is that the service will not be discovered - and we'll be able to investigate.

Additionally, we will be filling an upstream bug report to the Go project so they're aware of this issue and hopefully nobody else will have to learn about it the hard way.

Sorry for any inconvenience this caused! We're working around the clock to make sure that Discord is reliable and available for everyone, especially as utilization of the platform is at an all time high. Thank you for choosing Discord as your place to hang out and talk to your friends!

Originally printed at:
https://discord.statuspage.io/incidents/62gt9cgjwdgf

Imagining Your Post-Incident Report As a Documentary

By Jaime Woo

How you tell a story matters. We know this as the best stories feel like they arrive fully formed, spontaneous and unmanipulated. The worst ones have us regularly checking our phones for the time, and then questioning physics because *how has only four minutes passed*?

We may not think of post-incident reports as a form of storytelling, but they are an interpretation of the truth. They can't be the definitive truth, since, frankly, that's an impossibility. No matter how rigorous the investigation, there is simply no way to capture all the details to encapsulate the capital-T truth.

Every story has a point of view—even technical reports inject some subjectivity by which facts are included, the descriptive language used, and the order information is listed in. That these reports are subjective doesn't diminish their power; in fact, by accepting that we are telling a story, we can do our jobs of relating an incident better.

What happens when we, instead, embrace that the best thing we can do might be telling a story? What benefits come from imagining our reports as documentary? We think of documentary as the creative retelling of truth, a riff on the definition by John Grierson, who coined the term "documentary film." First, it underlines the distinction between actuality and any works we create about it; and, second, that in the act of building a report

we use human judgment: because we can't know everything that occurred, we capture what we can in an attempt to create and reconstruct what happened.

"To truly understand how an incident unfolded, you need to experience the incident from the perspectives of the people who were directly involved in it," notes Netflix's Lorin Hochstein on his blog. "Only then can you understand how they came to their conclusions and made their decisions." A helpful tip Lorin suggests is writing the narrative description in the first-person rather than from the third-person.

You might also consider asking world-building questions familiar to any documentarian:

- Who are your characters?
- What are their goals?
- What are their worries?
- What do they see, hear, and feel?

The right details build context, enrich understanding, and allow the reader to form connections between ideas. We always acknowledge the caveat that all narratives are speculative; still, building a stronger narrative in post-incident reports can do the seemingly magical task of transporting readers to the past to answer why the actions taken by those involved made sense at the time.

Check out Lorin's blog post at
https://surfingcomplexity.blog/2020/01/16/getting-into-peoples-heads-how-and-why-to-fake-it/

Etcetera

Thanks for checking out the Post-Incident Review.

You can find more of our writing at Morning Mind-Meld, a biweekly newsletter with thoughts on industry news.

Recently, we discussed what it meant to lose the hallway track in light of 2020 conferences being cancelled. Here's an excerpt:

"The hallway track is a place where you can learn from others like you, and hear conversations that are too private to share on social media. (You know, the ones that don't have to be polished and performed.) The hallway track is where you get to enjoy not just the ideas of the industry you're in, but the people. We've met our heroes through the hallway track, and strangers who instantly felt like friends we've known forever. These moments are necessarily ephemeral: that is the magic of live events."

Read the full article at https://morningmindmeld.com

Our version of an ad

Shout out to Justin Garrison (@rothgar), who shared his funniest experience getting paged: on Splash Mountain! Illustration by Denise Yu (@DeniseYu21).

A collection of public postmortems
for you to read, reflect, and annotate
wherever and whenever you'd like.

Visit us at the Post-Incident Review website
https://zine.incidentlabs.io

A Project by Incident Labs