Dan Slimmon

Evidence-oriented SRE

Dead air on the incident call

ON 2024/03/18 / BY DAN SLIMMON / IN INCIDENT-RESPONSE

When troubleshooting a high-impact software failure in a group, you have to be ready for shifts in tenor. One moment there's a frenzy of coordination, and the next: absolute silence.

Silence is natural and often useful. But to be an effective **incident commander** – whose job is to keep the problem-solving effort moving forward – you must develop a keen ear for silence.

Silence can mean different things to different people in different situations. In this post, I'll present a few incident scenarios and explore the role of the incident commander in breaking (or simply abiding in) dead air.

"Any minute now"

Sylvain (from the [s]upport team) has spun up an incident call after getting 3 consecutive reports of broken CSS on the production app. You are the incident commander, and Oscar (from [o]perations) has immediately volunteered to be the primary investigator. Deanna and Deepak (software [d]evs) are also on the call.

There's some ambiguity about whether the CSS issue merits a status page post. Nobody has found a "smoking gun" showing that, for example, 404 errors are happening at an elevated rate. And now Oscar announces, "I'm seeing some log entries from the web server that look a little weird. I'm gonna look at those." This is the beginning of a 5-minute silence.

During the silence, Deanna, Deepak, and Sylvain are all waiting, hoping that these log entries that Oscar just noticed turn out to be the smoking gun. They're putting their eggs in the basket of Oscar's intuition. Hopefully he's seen this issue before, and any minute now he'll say "Okay, I'm pushing a fix."

The problem is, it's equally possible that Oscar has latched onto a red herring (some salient but ultimately irrelevant observation). If there were a conversation in place of silence, then Deanna could be researching the error message, or Deepak might be able to immediately rule out the error as a cause of the broken CSS, or Sylvain could provide a detail from one of the customer tickets that would narrow the scope of the investigation. But instead, everybody's twiddling their thumbs hoping for Oscar to emerge with a fix.

An incident commander is responsible for keeping the whole problem-solving effort moving forward. So it's incumbent on you to interrupt this silence.

Try drawing more information out of Oscar:

- "Oscar, do you mind sharing your screen so Deepak and Deanna can see the weird log messages too?"
- "What's the error message, Oscar? Can you send a link to a log search?"
- "Do we know when these log events started? Does that line up with when we started receiving these support tickets, Sylvain?"

The more we audit each other's thought processes, the more effective we are at joint problem-solving. An IC must make this happen.

"LGTM"

Sylvain has spun up an incident call after getting 3 consecutive reports of broken CSS on the production website. You are the incident commander.

Oscar has checked a bunch of graph dashboards and hasn't found any evidence of a widespread system failure. He's said as much. Now there's relative silence on the call for five minutes.

Deanna and Deepak are basically in agreement with Oscar: there's no evidence of a system health issue. To them, and to Oscar, it's not really clear how strong a signal Sylvain has. It could just be a coincidence that these three reports all arrived in a row. The engineers on the call are thinking, *I guess we'll keep poking at this, but we're not even sure this is a real issue. We need more information.*

Sylvain, on the other hand, is positive that something is wrong. Getting 3 support tickets in a row about the same behavior is very strong evidence to him. He's presented his information to the investigators, and now he's thinking, *Okay, they say it's not a widespread issue. But I'm sure Oscar is getting to the bottom of it*.

There's been a common ground breakdown (https://blog.danslimmon.com/2015/10/19/troubleshooting-chatops-ddx/), and as a result, a silence that becomes more and more frustrating.

As incident commander, you should focus the group's attention on observable symptoms by asking questions like:

- "Has anybody been able to reproduce these broken page-loads in a browser? Preferably with Dev Tools turned on?"
- "Sylvain, I don't have an intuition for support ticket frequencies. How unusual is it to get 3 reports of the same thing right in a row like this?"
- "Can we find, in the access logs, just one example of a stylesheet request that returned a non-200 response?"

"Let's see here…"

Sylvain has spun up an incident call after getting 3 consecutive reports of broken CSS on the production website. You are the incident commander. The investigation has been going along, and Oscar is chasing down a hunch that a particular error message from the web server is related to the stylesheet failures. Deanna is digging into some code to help validate Oscar's hunch.

Deepak joins the call. There's no chatter, as everyone is waiting for Oscar and Deanna to come up with their findings. So Deepak reads the chat scrollback, which takes him about 5 minutes. It's not until the end of those 5 minutes that Deepak understands what Oscar and Deanna are working on.

As it happens, Deepak has seen the web server error message in question before. He knows what it means, and he can explain why it's a red herring. But for the 5 minutes it takes him to get up to speed by reading the chat scrollback, silence persists.

In order to keep a problem-solving effort moving forward, an incident commander should ensure that every new participant gets up-to-date knowledge of what the group is doing and why. At small scale (less than, say, 10 people on the call), you can do this verbally. For example, you could say to Deepak when he joins the call, "Hi Deepak. Right now, Oscar and Deanna are investigating a web server error message that might be related to failed stylesheet loads. You can see the error message in the chat."

When there are more than 10 people, the verbal approach stops working. It becomes necessary to have a shared document of some sort, continuously updated by a "scribe." It's not sufficient for this document to be merely a timeline of events: it must highlight the *current state* of the joint diagnostic

effort. I recommend clinical troubleshooting (https://blog.danslimmon.com/2024/03/08/clinical-troubleshooting-diagnose-any-production-issue-fast/) for this.

“I need 5 minutes”

When incident response is going right, everybody understands what’s being done by whom, and why. As information comes to light and our strategies evolve, it takes more or less constant communication to maintain this state. That’s why silence on an incident call is so often an indicator of trouble: when there’s silence, communication isn’t happening.

There is, however, a healthy kind of dead air.

Sometimes an investigator needs to go silent for a while to chase down a hunch, or collect some data, or research some question. As long as such a silence is negotiated in advance, with a specific time to reconvene, it can serve a crucial purpose. I call this **functional dead air**.

It’s the job of the incident commander to ensure that every nontrivial silence is functional. First, communicate what’s being done by whom, and why. Only then, do it.

DEVOPS INCIDENT-RESPONSE OPS SRE TROUBLESHOOTING

One thought on “Dead air on the incident call”

1. Pingback: Dead Air on the Incident Call nalgeon on March 18, 2024 at 19:18 Hacker News: Front Page - Bharat Courses