When things go wrong

Lea Kissner

@LeaKissner on Twitter

Plan for panic

- Write things down
- … because your adrenal system will be activated
    - Physical effects: pounding heart, trembling, etc.
    - Cognitive effects:
        - Interpreting things as overly negative
        - Feeling rushed
        - Reduced capacity to make good decisions
- So plan ahead
    - Have written procedures and playbooks
    - Consider hard choices
- Use physiological hacks: take a moment for everyone to breathe deeply

Incident steps

0. Assign an incident commander to coordinate and manage operations

1. Find the cut. What's going wrong? Who is affected? Incident, vulnerability, or false positive?

2. Stop the bleeding. Stop the bad thing from going wrong (not necessarily fix)

3. Clean up the blood. Fix whatever's broken. Make sure you understand the root cause(s). Put in place longer-term measures to avoid a repeat

Step 0: Assign a commander

- This person manages coordination and operations
  - ***Not*** doing the debugging
  - Make sure things are written down
- There can be only one.
  - Explicit handoffs
  - Name them in the coordination location: document, Slack channel, etc.
  - Consider a jaunty hat

Step 1: Find the cut

Questions to ask:

- What's going wrong? Is anything going wrong?
- Who is affected?
  - How many people?
  - What characteristics? Type of user (admin, employee, end user), geographic region, other characteristics
- Incident or vulnerability?
  - Vulnerability: a flaw in your systems or procedure which could allow a bad thing to happen
  - Incident: the bad thing actually happening
  - Remediation is different. Legal obligations are different.

Step 1: Find the cut

Human issues

- Employee errors. People mess up. Your remit covers everything, not just production systems.
- User says “I told it to do X and it did Y!”
    - Sometimes they didn’t read
    - Sometimes they forgot
    - These are technically false-positives, but they are also invaluable to making the product work
    - … and sometimes those users are right and your system is broken
        - Maybe you asked for the wrong consent… or asked for different consents… or updated the product… or didn’t ask the people already using the product

Step 1: Find the cut

Bug issues:

- Alice gets Bob’s data… sometimes
    - Thread safety, race conditions
    - Cache collisions
    - Memory corruption
- The wrong “correct” behaviour is happening
    - Sometimes a logic error
    - Sometimes a protocol error: versioning, type error
- ML model doesn’t do what it’s supposed to
    - … you’re using ML
    - A lot of the time it comes down to bad signals, biased dataset (sometimes in very subtle ways), or insufficient data
    - You should be testing better but still prepared not to catch every issue

Step 2: Stop the bleeding

- Stop the bad thing from going wrong
  - This is not necessarily the same thing as fixing the thing!
  - Design for quick stops:
    - Take down the service
    - Binary rollback
    - Selective blocking of network requests
    - Feature flagging
  - Messy or partial fixes are often better than no fix (even if you have to clean them up right away)
- Remember, you can’t un-leak someone’s data, but you also cause users more harm by being noisy
  - … think about your choices in advance

Step 3: Clean up the blood

- Short-term remediation
  - Fixes the problem
  - … but the system may still be in a brittle state
  - Example: had a surprise cookie, remove the cookie
- Longer-term remediation
  - Get from a not-broken state into a good state
  - Example:
    - Scan for other unexpected cookies
    - Check code, scan 3p code, scan add-ons before they're added in future
    - Make active guards against setting cookies which are not allowed
  - People will put off remediation
    - Align incentives so the team who will fix the problem feels the problem
    - Have the VP sign in blood that they'll get it fixed by a certain time

Step 3: Clean up the blood

- Postmortem to learn: what went well, what went poorly, what happened, why
- Goal: To avoid this incident and related classes of incident
- What to do:
  - Look for underlying causes. Ask “what caused this” at least 3x. Go deeper.
  - Blameless. Because otherwise people hide.
  - Look for related issues.
  - Fix or guard against issues:
    - Fix bugs
    - Harden systems: avoid failure modes or make active guards against failures
    - Reduce access
    - Better monitoring and alerting
    - Training: make sure your privacy and security teams know, but somewhat brittle
  - Improve incident response process

Thanks!

Lea Kissner - *@LeaKissner (Twitter)*

Sample incident document: https://buildwithrespect.com/2020/09/30/pepr-2020/

Bonus content in my slides:
https://www.usenix.org/conference/pepr20/presentation/kissner

Bonus content: metrics to track

- They can tell you valuable things. They can also mess you up big-time.
- Things to track:
  - How many reports are you getting? What breakdown of false positive/vuln/incident? How long are they taking to resolve? Breakdown of root causes.
- Goals here are:
  - Make sure you're staffing appropriately. It's very easy to burn out people handling incidents! They need to be able to tap out and and off.
  - Make sure you're investing sufficiently in the right kinds of hardening, e.g. moving to more-automated systems to prevent manual errors, bug-scanning, better UX
  - Having the same issues over and over again is both bad for users and dispiriting for the privacy folks. We'll never be free of all errors, but let's at least have them be new and hopefully less bad for users.

Bonus content: metrics to avoid

- You need to be careful about metrics
  - Measuring something will make people want to change it
  - … but their way of changing them can be counterproductive
- For example, if you're seeing a lot of incident reports from one product and very few from another, that could mean:
  - One product has a lot more bugs, is badly designed, or more brittle
  - One product is doing something with more privacy and/or security exposure, like having a sharing function and a recommendation engine vs a product where it only shows your information back to you and doesn't use ML
  - One product team/external folks is doing a better job of reporting potential incidents
- … one of these things is *much* easier to change than the other two. And it's not the one you want.