Zero Touch Prod

Towards Safer and More Secure Production Environments

SREcon 2019 EMEA

Michael Czapiński (mcz@google.com)
Rainer Wolafka (rwolafka@google.com)
Borg SRE Zurich

Agenda

- Motivation
- Reliable Automation
- Safe Proxies
- Adoption
- Conclusion (with Q&A)

Motivation

Outage Scenario 1 *"I never make tipos"*

- SRE wants to bring down a service for maintenance in a cluster without user facing traffic (drained cluster) but accidentally affects a production cluster
    - **Typo** in the cluster name
    - **Cut and paste error** (wrong command in the clipboard)
    - **Wrong terminal** window
    - *<insert your worst nightmare here>*

Outage Scenario 2 *"With great SRE power comes great responsibility"*

- Malicious SRE (or "attacker") wants to cause harm by deliberately bringing down or disrupt a service
  - SREs know how to run a service but also have the knowledge to bring it down
  - Compromised SRE privileges and credentials

Outage Scenario 3 *"What can possibly happen?"*

- SRE wants to "just" reboot some servers and delete old data. Writing an ad-hoc and unreviewed script to SSH into these machines will do the job.
  - Mistake in the script with wildcards and variable substitution
    - *→ affects all machines and is applied to other data repositories on the system*
  - No rate limiting. No review/approval or proper change control
    - *→ takes effect immediately*
    - *→ unilateral change and no traceability*

Zero Touch Prod (ZTP) to the rescue

- **Goal:** Make production safer and prevent outages
- **Rule:** Every change in production must either be
    - made by **automation** (no humans)
    - **prevalidated** by software
    - made via an **audited break-glass mechanism**
- ZTP encompasses a set of **principles** and **tools** to enforce this

How much is it worth? An attempt to quantify

- Establish criteria for outage classification (in scope for ZTP)
- Analyze a statistically relevant set of post mortems using this criteria
- Extrapolate the findings to all post mortems
- Results (for our case)
    - **~13% of all outages in scope could have been prevented/mitigated with ZTP**
    - If your organization has an estimated average outage cost this translates to $$$
    - In our case the "outage savings" were significantly higher than the ZTP adoption investment

Reliable Automation

Limiting Privilege: Authority Delegation

- Allow systems to only perform actions they are required to do. Nothing more!
    - Similar to the "principle of least privilege" in information security
- Example
    - Automation system that wants to manage production resources on behalf of its users (such as automatically actuating an intent in production)
    - This system should be able to only manage specific production assets with tightly controlled access

Enforce safety policies: Safety Checks

- Establish a safety check system to approve changes to production infrastructure
    - Critical systems should consult it before making production changes
    - Can be used to enforce a global production freeze in the case of an outage
- Examples
    - Prevent stopping a service if there is user facing traffic configured for it
    - Coordinate operations originating from multiple systems to a specific part of the production infrastructure, e.g. prevent rebooting a significant part of the fleet at the same time

Controlling the rate of change: Rate Limiting

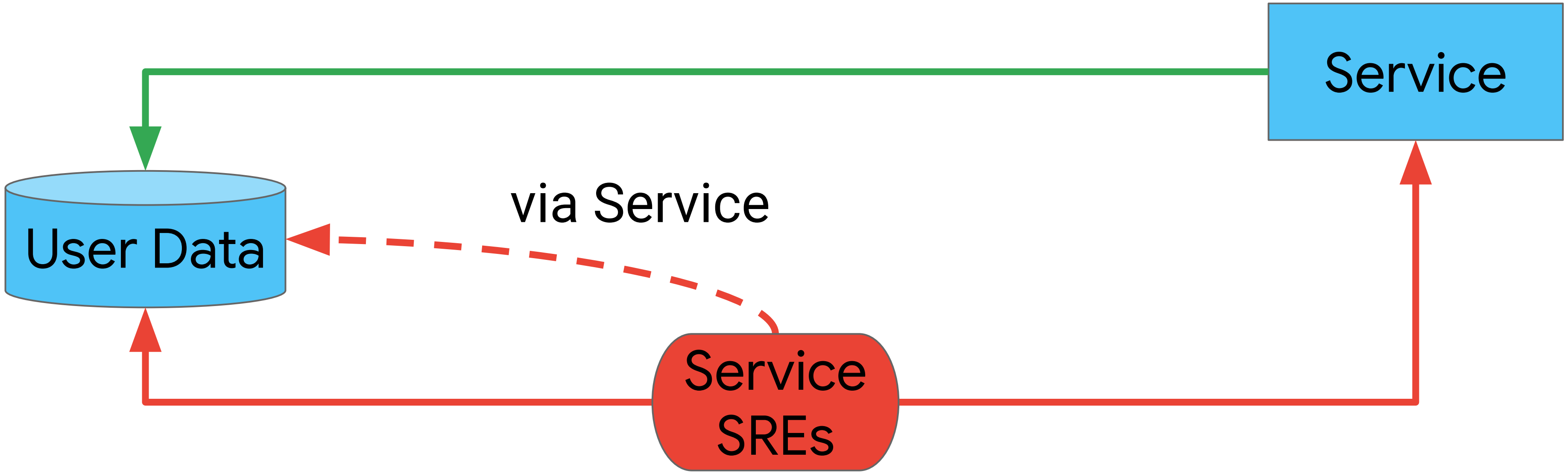
- Control and if needed throttle changes to production to an "acceptable rate"
  - Do not change all of production "at once"
  - Limit the potential immediate damage to a subset of production
- Examples
  - Installing new Software on a production system
  - Rebooting servers (maintenance)
  - Pushing a configuration change

Safe Proxies

With great power comes great responsibility...

- turning service up and down
- pushing configurations
- capacity planning
- access to user data
- **...and great temptation: all those powers can be used by a malicious actor**

Initial service access setup



Access is granted via direct membership in ACLs

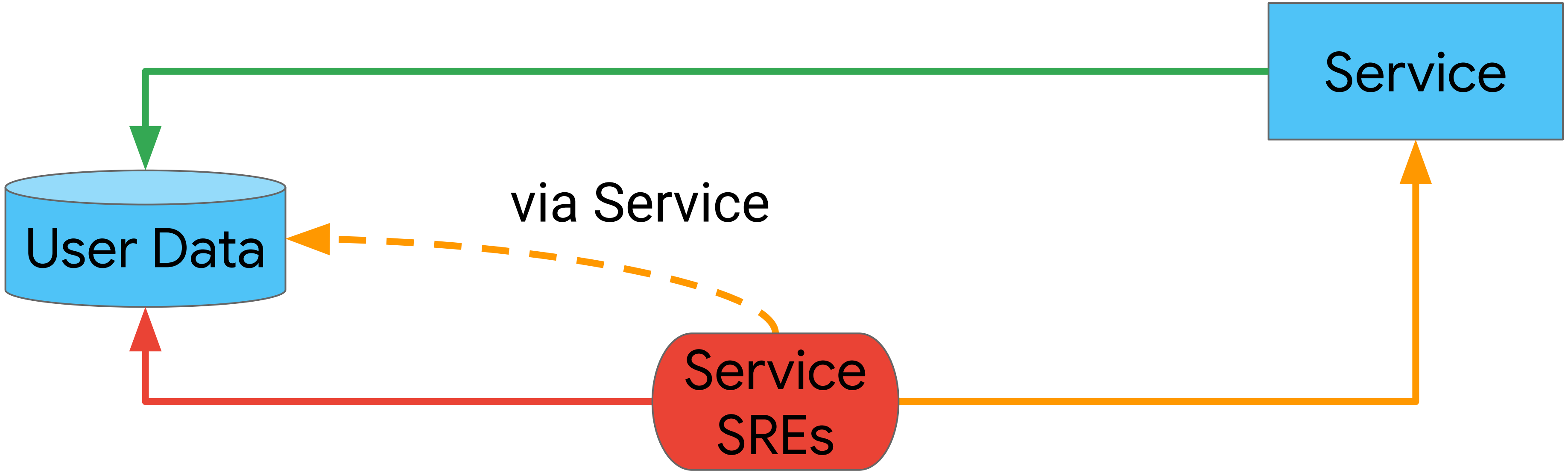
Assumptions

- 14 people per healthy SRE oncall rotation
  - 2 oncallers at a time, oncall 1/3 of the time; 2 sites, N+1 per site
- Number of people with potentially dangerous access scales linearly
  - 100 services => 1,400 SREs
- If every SRE makes a mistake leading to a major outage with 0.1% probability, then chance of at least one major outage is:
  - less than 1.4% for a team of 14 SREs
  - **more than 75% at 1,400 SREs**

Job control system can verify software provenance

- **Problem: SREs can run any binary with any configuration**
- **Solution: job control system should verify software provenance, i.e. that**
  - binary was built verifiably (i.e. we can trace the source of the code)
  - the source code comes from specific locations in the repository
  - all the source code was peer-reviewed
  - job configuration (flag values, config files etc.) was peer-reviewed
- **Why is this not enough?**
  - SREs can **unilaterally** turn the job down or revert to old version with known security issue

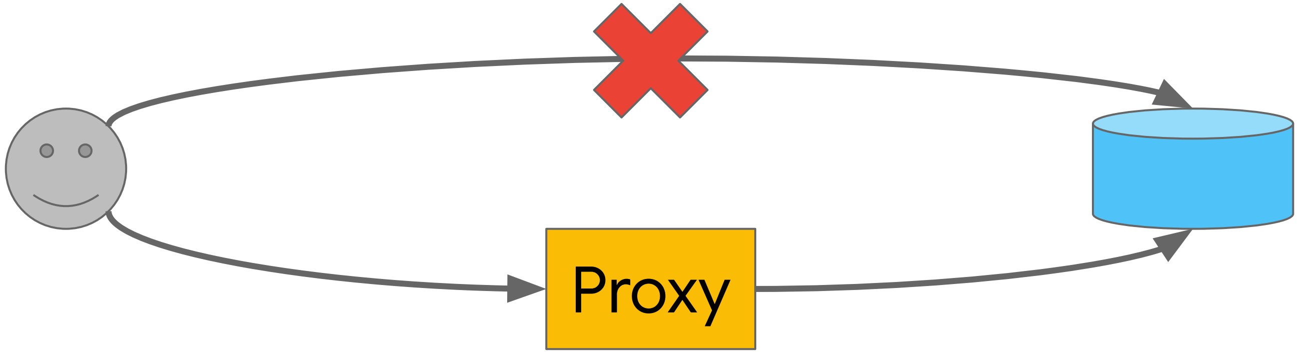
Service access is now partially protected



Google

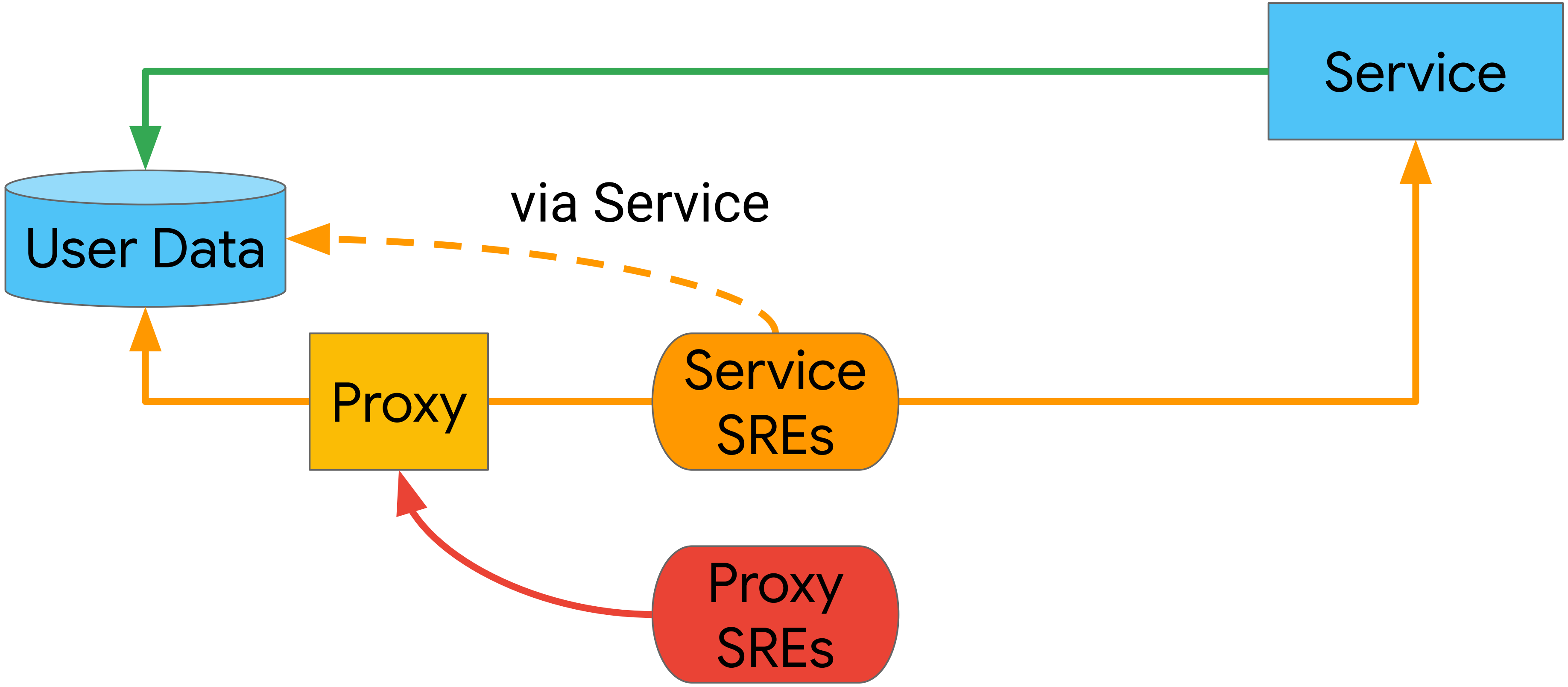
Use Safe Proxies to control access to user data

- Full audit log (who, when, what, **why**)
- Fine-grained authorisation: limited scope of user data queries
- Rate-limiting (prevents scraping or modifying user data at scale)
- **Two proxy types: RPC and CLI**



Google

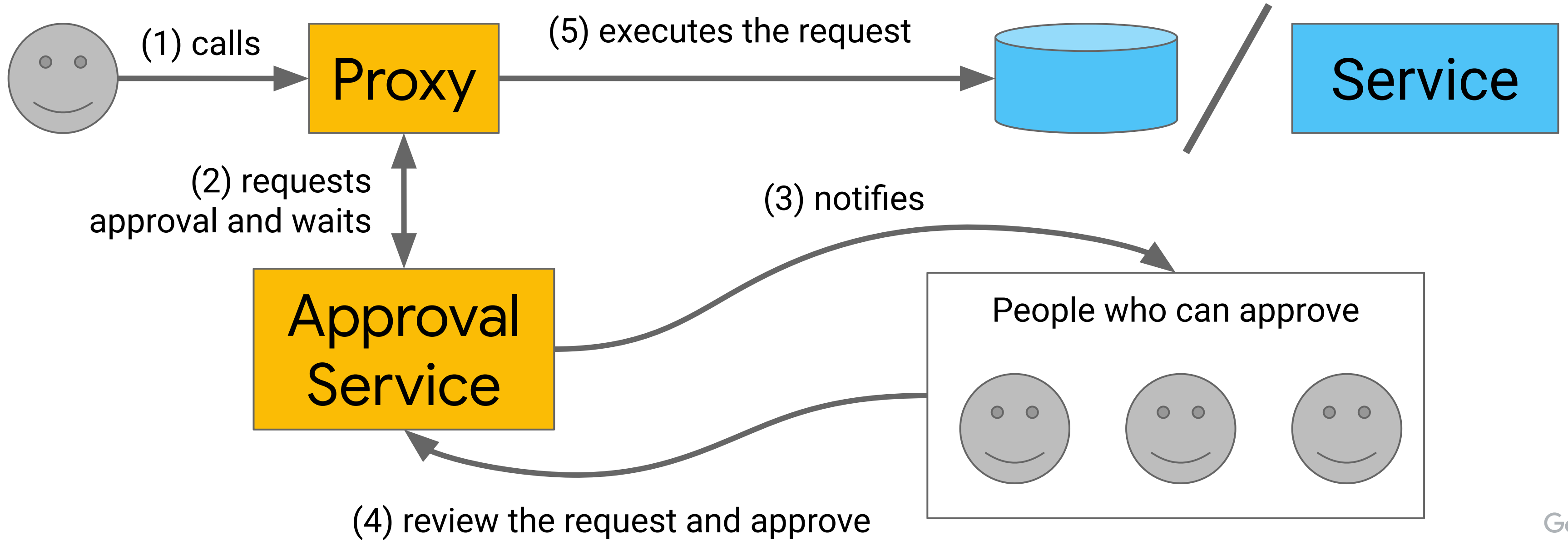
User data is partially protected (but access is unilateral)



Google

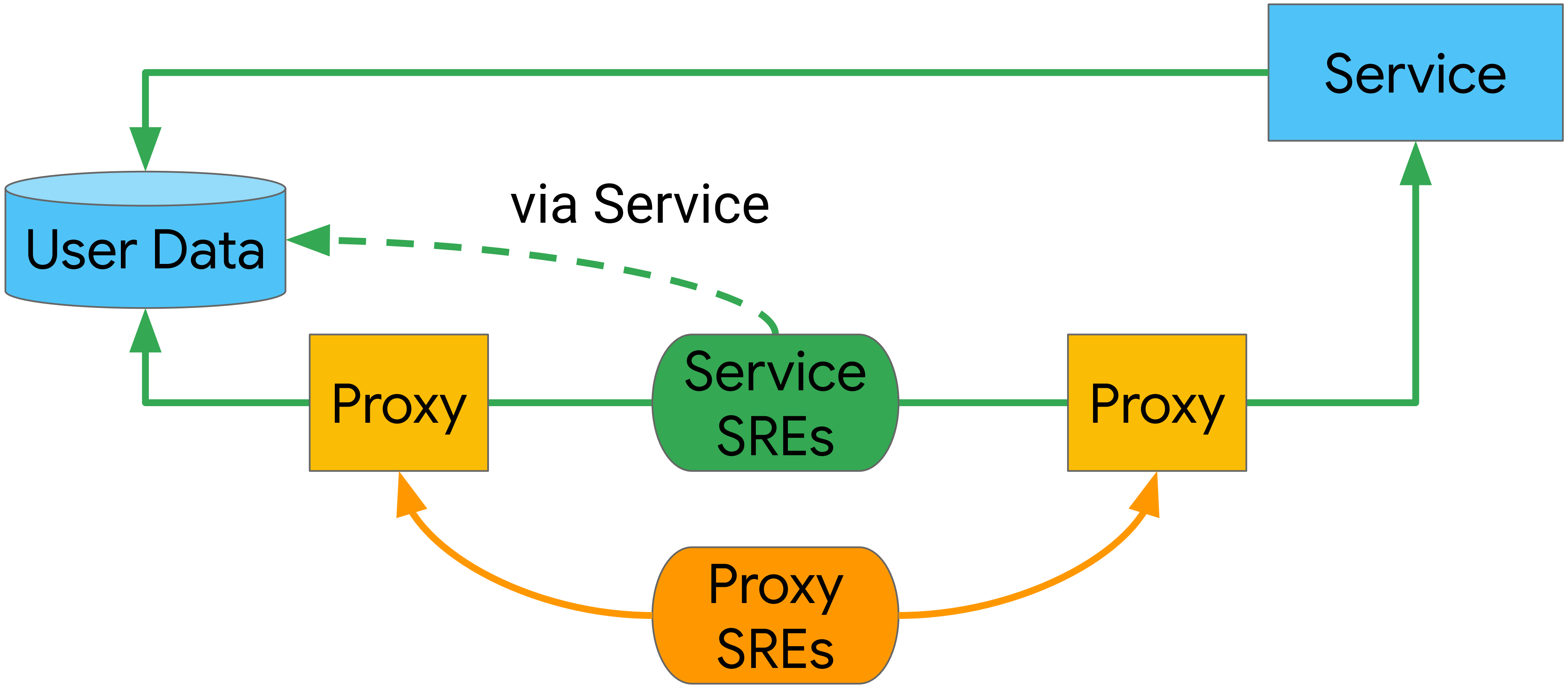
Unilateral access is convenient but potentially dangerous

- We require peer reviews for code changes; **why not production changes**?
- **Proxies can make service changes and user data access non-unilateral**



Google

Service and user data fully protected (no unilateral access)



What could possibly go wrong?

- There are new moving parts: **we need break-glass for emergency response**
- **Approval Service is unavailable:**
  - escalate to Approval Service SRE
  - request unilateral operation with justification (this should likely alert security people to review)
- **Some proxy dependencies are unavailable:**
  - escalate to dependency service SRE or use its break-glass mechanism
  - escalate to Proxy SRE to enable break-glass mode (e.g. store audit logs locally)
- **Proxy is unavailable:**
  - request temporary membership in relevant production groups (initial service access setup)

Adoption

How to support ZTP adoption across an organization ?

- Establish a framework for assessing and tracking adoption
    - Define a set of metrics that classify your production safety criteria
        - Examples: Number of humans in production groups, frequency of using emergency procedures (for non emergencies), number of safety check denials, …
    - Map these (many) metrics to a "ZTP level" or "production maturity" model
    - Provide dashboards to assess, track and monitor compliance
    - Identify areas of non compliance

Conclusion

Recap

- Humans make mistakes (repeatedly): don't let them
  - Automation makes mistakes too but we claim this is easier to fix
  - Automation has to be fixed only once whereas humans join and leave a team all the time
- Follow a set of principles to enforce production safety practices
  - Reliable automation (Authority Delegation, Safety Checks, Rate Limiting)
  - Safe Proxies
- Provide a framework to assess and track compliance with ZTP principles

Recap - Benefits from using safe proxies

- Full audit log (who, when, what, why)
- Fine-grained authorisation: limited scope of user data queries
- Rate-limiting
- Removes unilateral privileged access:
    - **Reduces the risk of an outage due to accidental production change**
    - **Reduces the risk of unauthorized access to user data**

“Every change in production must be either made by **automation, prevalidated by Software** or made via **audited break-glass mechanism.**”

Seth Hettich

Former Production TL, Google