Go Fast Without Breaking Things

Robert Ross, CTO & Founder
rross@curtail.com
February 13, 2020

For a business to function, its systems must stay up and running...
Sponsor Message: How ReGrade Works

2 Minute Explainer Video

Link is safe, can be accessed here  https://youtu.be/QkrbgBQyBPI
Agenda

• Code Updates Gone Wrong
  • Cloudflare / Cloudbleed
  • Facebook / Exposed Admins
• Existing Processes and Tools
• What’s Missing?
• Q&A
Who is Cloudflare?

- CDN: Content Delivery Network
- Caching of static HTTP content at edge servers
- Geographically distributed
- Load content from customer websites, deliver to end users
- $250M revenue
- ~1200 employees
- Over 2 million customers and over 20 million internet properties
- CI/CD, Performance Monitoring, Log Monitoring, etc.
What was Cloudbleed?

- **22 September 2016**: Upgrade to HTML parser code was deployed
- **All QA tests passed** before release
- Log and performance monitoring showed no unusual activity
- **13 February 2017**: Flaw expanded in scope, affecting more sites
- **17 February 2017**: Google’s Project Zero notifies Cloudflare of bug
- With knowledge of where to look, bug is quickly identified
- **18 February 2017**: Bug is fixed (149 days later!)
Example Cloudbleed Leak
Cloudbleed
Consequences

- HTTP cookies (including session cookies)
- HTTP headers (including authentication and CSRF tokens)
- HTTP POST bodies (usernames, passwords, personal information)
- Data was stored by search engines during the 149-day window
- Cloudflare coordinated with search engines to purge leaked data
Who is Facebook?

• The Social Network
• 1.6 Billion Daily Active Users
• $70 Billion 2019 revenue
• 45,000 employees
• CI/CD, Performance Monitoring, Log Monitoring, etc.
What happened with Facebook Pages?

- Facebook Pages allows companies and other organizations to have a presence on Facebook without being linked to any one individual
- Individuals that contribute to a Page are anonymous by default
- **9 January 2020:** Code update pushed live
- **All QA tests were passing before release**
- Log and performance monitoring showed everything was healthy
- **10 January 2020:** Facebook is notified by researcher and fixes code

*Facebook says the bug was the result of a code update that it pushed Thursday evening. It’s not something most people would have encountered on their own, since it took navigating to a Page, viewing an edit history, and realizing that there shouldn’t be a name and profile picture assigned to edits to exploit it.* – Lily Hay Newman, WIRED
Facebook Pages Consequences

“... despite the Friday morning fix, screenshots circulated on 4chan, Imgur, and social media appearing to show the accounts behind the official Facebook Pages of the pseudonymous artist Banksy, Russian president Vladimir Putin, former US secretary of state Hillary Clinton, Canadian prime minister Justin Trudeau, the hacking collective Anonymous, climate activist Greta Thunberg, and rapper Snoop Dogg, among others.” -- Lily Hay Newman, WIRED

“For sensitive Pages, I would not rule out that some people may be feeling that they are in danger due to what happened today”
-- Lukasz Olejnik, Research Associate at Oxford University’s Center for Technology and Global Affairs
Existing Process

- Developer Testing
- QA Testing
- Static Analysis
- Deployment
- Logging
- Runtime Application Self Protection
- Performance Monitoring
- Resource Monitoring
- Customer Feedback
- Rollbacks
Developer and QA Testing

• Developers are often limited on time for a release

• If developers could anticipate every bug in test, there would be no bugs

• QA performs specific, discrete tests which check all anticipated failures

• Many basic errors are caught at this stage

• Many failures are not anticipated and are missed
Static Analysis

- Static Application Security Testing (SAST)
- Lint (1978): The Original
- Vulnerability databases
- File system (including container) hash databases
- Scanning for known vulnerable components only helps if the vulnerability is known, a patch is available, and resources are available to update the software with the patch.
- Static analysis is limited by Rice’s Theorem:

Rice’s theorem states that all non-trivial, semantic properties of programs are undecidable. A semantic property is one about the program's behavior (for instance, does the program terminate for all inputs), unlike a syntactic property (for instance, does the program contain an if-then-else statement). A property is non-trivial if it is neither true for every computable function, nor false for every computable function.

--Wikipedia
Deployment Models for Risk Mitigation

• Rolling Deployments
  • Limit fallout when things break
• Feature Flags
  • Choose who is affected when they break
• Blue/Green Deployments
  • Have a fallback plan ready when things break

The purpose of these models is to limit (NOT avoid) the damage caused by software flaws. Failures in production are often accepted as unavoidable.
Runtime Application Self Protection

- Intercept/hook key functions
- Catch common anti-patterns
- Address space layout randomization
- Data execution prevention
- Distinguish *presence* of vulnerable code versus *actual* vulnerability

Great for some classes of vulnerabilities (buffer overflow, SQL injection) while completely missing other significant defects
Performance, Resource, and Log Monitoring

Good at catching systemic failures quickly after deployment

• Resource exhaustion

• Degraded infrastructure

• Service outages

• Common configuration errors

Many significant failures are invisible to these metrics. Significant flaws may have no impact whatsoever on resource utilization or performance.

Log monitoring, much like developer testing, relies on developers anticipating potential problems and logging relevant details. If developers could always anticipate every failure there would be no bugs.
Customer Feedback, Bug Bounties, and Rollbacks

These are often the last resort for companies that have missed flaws at the earlier stages.

- Customers experiencing problems complain to support
- Security researchers discover flaws and claim a bug bounty
- Various tools and processes exist enabling rapid rollback when new software is causing problems

The problem has already happened. Even if it’s fixed quickly it may still become a front-page headline and erode customer trust. It’s nice to reduce the number of irate customers, but wouldn’t it be better to keep them happy and save on the cost of bug bounties?
What’s Missing?

• Actual network behavior is not measured
• Logs and performance monitoring often miss important bugs
• Deployment model mitigations split populations, forcing statistical rather than direct comparison
• Identified flaws can be difficult for developers to reproduce
Directly Compare Network Behavior

Production Customers -> HTTP/S -> Passive Sidecar

V1.0 <-> HTTP/S <-> V2.0

V2.0 <-> SQL <-> DB

V1.0 <-> SQL <-> DB
Advantages of Comparing Behavior

- Flaws can be identified easily and precisely before risking customer experience
- Provide developers with ability to replay observed activity and reproduce bugs
- Find problems that QA did not anticipate identify ALL meaningful differences
- Discover errors not anticipated by discrete QA tests
- Measure what’s important: logs are not the most accurate representation of customer experience
Go Fast Without Breaking Things

- Current software development process often misses subtle or unanticipated flaws in new software releases
- Many organizations have invested in tools and process that allows rapid detection and response when things break in production, but at this point it’s already too late
- New ways of testing before releasing code enables the detection of these issues earlier in the software lifecycle, avoiding the need to respond to an emergency after the fact
- Shift Left: Detecting flaws earlier in the process allows organizations to save time and money, protect their reputation, and deliver more secure and reliable software at a faster pace
Advantages of Comparing Behavior

We help companies dramatically reduce the % of software change failures that cause costly rollbacks and poor customer experience.

Unlike typical after-the-fact and incomplete logging and monitoring tools, we compare actual network behavior.

We discover and accurately reproduce flaws before they “go live”.