CHAPTER 19

Case Study: Chrome Security Team

By Parisa Tabriz
with Susanne Landers and Paul Blankinship

In the early days of Google, security work—including product-focused security—was entirely centralized. Chrome was one of the first products to build out a security-focused organization, to tackle the unique challenges of building a secure, modern web browser. This case study describes the evolution of Chrome's security team, the core principles it developed, and some concrete ideas about how to scale security across an organization.

Background and Team Evolution

In 2006, a team was formed at Google with the aim of building an open source Windows browser, in less than two years, that would be more secure, faster, and more stable than the alternatives in the market. This was an ambitious goal and presented unique security challenges:

- Modern web browsers have similar complexity to an operating system, and much of their functionality is considered security-critical.
- Client-side and Windows software were different from most of Google's existing product and system offerings at the time, so limited transferable security expertise was available within Google's central security team.
- Since the project intended to begin and remain predominantly open source, it had unique development and operational requirements and could not rely on Google's corporate security practices or solutions.

This browser project ultimately launched as Google Chrome in 2008. Since then, Chrome has been credited as redefining the standard for online security and has become one of the world's most popular browsers.

Over the past decade, Chrome's security-focused organization has gone through four rough stages of evolution:

Team v0.1
: Chrome did not formally establish a security team before its official launch in 2008, but relied on expertise distributed within the engineering team, along with consulting from Google's central security team and third-party security vendors. The initial launch was not without security flaws—in fact, a number of critical buffer overflows were discovered within the first two weeks of public availability! Many of the initial launch bugs fit the pattern of flaws that result from developers under time pressure trying to ship C++ code that's optimized for performance. Bugs also existed in the implementation of the browser application and web platform. Discovering bugs, fixing them, writing tests to prevent regressions, and eventually designing them away is part of the normal process of a maturing team.

Team v1.0
: A year after the public beta release, with actual usage of the browser beginning to grow, a dedicated Chrome security team was created. This initial security team, composed of engineers from Google's central security team and new hires, leveraged best practices and norms established at Google and also brought in new perspectives and experiences from outside the organization.

Team v2.0
: In 2010, Chrome launched a Vulnerability Reward Program (VRP) to recognize contributions from the larger security research community. The overwhelming response to the VRP announcement provided a useful incubator in the early days of the security team. Chrome was originally based on WebKit—an open source HTML rendering engine that previously hadn't seen much security scrutiny—so one of the team's first missions was to respond to the huge influx of external bug reports. At the time, Chrome's engineering team was very lean and not yet familiar with all of the WebKit codebase, so the security team found that the most expedient approach to getting a vulnerability resolved was often to just dive in, build up expertise on the codebase, and fix many of the bugs themselves!

  These early decisions ended up having a big impact on the team's culture going forward. It established the security team not as isolated consultants or analysts, but instead as a hybrid engineering team of security experts. One of the strongest advantages of this hybrid approach is in the unique and practical insights it provides about how to incorporate secure development into the day-to-day processes of every engineer working on Chrome.

Team v3.0

By 2012, Chrome usage had grown further, as had the team's ambitions—and the attention from attackers. To help scale security across the growing Chrome project, the core security team established, socialized, and published a set of core security principles.

In 2013, after bringing on an engineering manager and hiring more engineers dedicated to security, the team held an offsite meeting to reflect on their work, define a team mission, and brainstorm about the larger security problems they wanted to tackle, along with potential solutions. This mission-defining exercise resulted in a statement that articulated the shared purpose of the team: to provide Chrome users with the most secure platform possible to navigate the web, and generally advance security on the web.

At that 2013 offsite session, to brainstorm in an inclusive way, everyone wrote their ideas down on Post-it notes. The team collectively clustered ideas to identify themes, which resulted in establishing a few evergreen focus areas of work. These focus areas include the following:

Security reviews

The security team regularly consults with other teams to help design and assess the security of new projects and review security-sensitive changes to the codebase. Security reviews are a shared team responsibility and help promote knowledge transfer. The team scales this work by writing documentation, hosting security training, and serving as owners for the security-critical parts of Chrome's code.

Bug finding and fixing

With millions of lines of security-critical code and hundreds of developers around the world constantly making changes, the team invests in a range of approaches to help everyone find and fix bugs as quickly as possible.

Architecture and exploit mitigation

Recognizing that you can never prevent all security bugs, the team invests in secure design and architecture projects to minimize the impact of any single bug. Since Chrome is available across popular desktop and mobile operating systems (for example, Microsoft Windows, macOS, Linux, Android, and iOS), which are themselves continually evolving, this requires ongoing OS-specific investments and strategies.

Usable security

However confident the team can be that Chrome software and the systems used to build it are invulnerable to attack, they still need to take into account how and when users (who are, after all, fallible) themselves make security-sensitive decisions. Given the wide range of digital literacy among browser users, the team

invests in helping users make safe decisions as they browse the web—making security more *usable*.

Web platform security
: Beyond Chrome, the team works on advancing security for developers who are building web apps so that it's easier for anyone to build safe experiences on the web.

Identifying accountable leads for each focus area, and later dedicated managers, helped establish a more scalable team organization. Importantly, the focus area leads embraced agility, team-wide information sharing, and project swarming or collaboration, so that no individual or focus area became siloed from other focus areas.

Finding and retaining great people—individuals who care about the team's mission and core principles and collaborate well with others—is critical. Everyone in the team contributes to hiring, interviewing, and providing ongoing growth-oriented feedback to their teammates. Having the right people is more important than any organizational detail.

In terms of actually finding candidates, the team has heavily leveraged its personal networks, and constantly works to nurture and grow those networks with people from diverse backgrounds. We've also converted a number of interns to full-time employees. Occasionally, we've reached out cold to individuals who have spoken at conferences or whose published work shows they care about the web and building products to work at scale. One advantage of working in the open is that it means we can point prospective candidates to details of our team's efforts and recent accomplishments on the Chromium developer wiki so they can quickly understand more about the team's work, challenges, and culture.

Importantly, we have pursued and considered individuals who were *interested* in security, but whose expertise or accomplishments were in other areas. For example, we hired one engineer who had an accomplished SRE background, cared deeply about the mission of keeping people safe, and was interested in learning about security. This diversity of experience and perspectives is broadly recognized as a key factor in the team's success.

In the following sections, we share more insights into how Chrome's core security principles have been applied in practice. These principles remain as relevant to Chrome today (ca. 2020) as they did when first written in 2012.

Security Is a Team Responsibility

One of the key reasons that Chrome has such a strong security focus is that we've embraced security as a core product principle and established a culture where security is considered a team responsibility.

Although the Chrome security team has the privilege of focusing almost entirely on security, team members recognize they can never own security for all of Chrome. Instead, they make an effort to build security awareness and best practices into the daily habits and processes of everyone working on the product. System design conventions aim to make the easy, fast, and well-lit path the secure path as well. This has often required additional work up front but has resulted in more effective partnerships in the long term.

One example of this in practice is the way the team approaches security bugs. All engineers, including security team members, fix bugs and write code. If security teams only find and report bugs, they may lose touch with how hard it is to write bug-free code or fix bugs. This also helps mitigate the "us" versus "them" mentality that sometimes arises when security engineers don't contribute to traditional engineering tasks.

As the team grew beyond its early days of vulnerability firefighting, it worked to develop a more proactive approach to security. This meant investing time in building and maintaining a fuzzing infrastructure and tooling for developers that made it faster and easier to identify changes that introduced bugs and to revert or fix them. The faster a developer can identify a new bug, the easier it is to fix, and the less impact it has on end users.

In addition to creating useful tooling for developers, the team creates positive incentives for engineering teams to do fuzzing. For example, it organizes annual fuzzing contests with prizes and creates fuzzing tutorials to help any engineer learn how to fuzz. Organizing events and making it easier to contribute helps people realize they don't need to be a "security expert" to improve security. The Chrome fuzzing infrastructure started small—a single computer under an engineer's desk. As of 2019, it supports fuzzing across Google and the world.[1] In addition to fuzzing, the security team builds and maintains secure base libraries (e.g., the safe numerics library), so that the default way for anyone to implement changes is the safe way.

Security team members often send peer bonuses or positive feedback to individuals or their managers when they notice someone modeling strong security practices. Since security work sometimes goes unnoticed or is not visible to end users, taking extra effort to recognize it directly or in a way that's aligned with career goals helps set up the positive incentives for better security.

Independent of tactics, if organizations don't already hold the position that security is a core value and shared responsibility, more fundamental reflection and discussion are needed to prove the importance of security and reliability to an organization's core goals.

1 Google open sourced ClusterFuzz, the fuzzing backend for its OSS-Fuzz service, in early 2019.

Help Users Safely Navigate the Web

Effective security should not depend on the expertise of any end user. Any product with a large-scale user base needs to carefully balance usability, capability, and other business constraints. In most cases, Chrome aims to make security nearly invisible to the user: we update transparently, we bias toward safe defaults, and we continually try to make the safe decision the easy decision and help users avoid unsafe decisions.

During the Team v3.0 phase, we collectively acknowledged that we had a cluster of open problems with usable security—problems that stem from humans interacting with software. For example, we knew users were falling victim to social engineering and phishing attacks, and we had concerns about the effectiveness of Chrome's security warnings. We wanted to tackle these problems, but we had limited human-centered software expertise in the team. We decided that we needed to strategically hire for more usable security expertise and serendipitously connected with an internal candidate who was interested in a new role.

At the time, this candidate was on the research scientist job ladder, from which Chrome had no precedent for hiring. We convinced leadership to hire the candidate, despite early reservations, by underscoring how the candidate's academic expertise and diverse perspectives were actually an asset to the team and necessary to augment its existing skill set. Partnering closely with the user experience (UX) team, with which security had occasionally been at odds in the past, this new addition to our team went on to establish Chrome's usable security focus area. Eventually, we hired additional UX designers and researchers to help us more deeply understand users' security and privacy needs. We learned that security experts, given their high comprehension of the way computer systems and networks work, are often blind to many of the challenges users face.

Speed Matters

User safety depends on quickly detecting security flaws and delivering fixes to users before attackers can exploit them. One of Chrome's most important security features is fast, automatic updates. From early days, the security team worked closely with technical program managers (TPMs), who established Chrome's release process and managed the quality and reliability of each new release. Release TPMs measure the rates of crashes, ensure timely fixes for high-priority bugs, roll releases forward carefully and incrementally, push back on engineers when things are going too fast, and hustle to get reliability- or safety-improving releases out to users as fast as is reasonable.

Early on, we used the Pwn2Own and later Pwnium hacking contests as forcing functions to see if we could actually release and deploy critical security fixes in under 24 hours. (We can.) This required strong partnership and a significant amount of help

and buy-in from the release TPM team, and though we demonstrated the capability, we've rarely needed to use it, thanks to Chrome's investment in defense in depth.

Design for Defense in Depth

No matter how fast the team is able to detect and fix any single security bug in Chrome, these bugs are bound to occur, particularly when you consider the security shortcomings of C++ and the complexity of a browser. Since attackers are continually advancing their capabilities, Chrome is continually investing in developing exploit mitigation techniques and an architecture that helps avoid single points of failure. The team has created a living color-by-risk component diagram so anyone can reason about Chrome's security architecture and various layers of defense to inform their work.

One of the best examples of defense in depth in practice is the ongoing investment in sandboxing capabilities. Chrome initially launched with a multiprocess architecture and sandboxed renderer processes. This prevented a malicious website from taking over a user's whole computer, which was a significant advancement in browser architecture for the time. In 2008, the largest threat from the web was a malicious web page using a browser compromise to install malware on the user's machine, and the Chrome architecture successfully brought that problem under control.

But computer usage evolved, and with the popularization of cloud computing and web services, more and more sensitive data has moved online. This means that cross-website data theft could be as important a target as compromise of the local machine. It wasn't clear when the focus of attacks would shift from "renderer compromise that installs malware" to "renderer compromise that steals cross-site data," but the team knew the incentives were there to make the move inevitable. With that realization, in 2012 the team embarked on Site Isolation, a project to advance the state of sandboxing to isolate individual sites.

Originally, the team predicted the Site Isolation project would take a year to complete, but we were off by more than a factor of five! Estimation mistakes like this tend to put a bull's-eye on a project's back from upper management—and with good reason. The team regularly articulated to leadership and various stakeholders the defense-in-depth motivation for the project, its progress, and the reasons why it was more work than first anticipated. Team members also demonstrated a positive impact on overall Chrome code health, which benefited other parts of Chrome. All of this gave the team additional cover to defend the project and communicate its value to senior stakeholders over the years until its eventual public launch. (Coincidentally, Site Isolation partially mitigates speculative execution vulnerabilities, which were discovered in 2018.)

Since defense-in-depth work is less likely to result in user-visible changes (when done right), it's even more important for leadership to proactively manage, recognize, and invest in these projects.

Be Transparent and Engage the Community

Transparency has been a core value for the Chrome team since the start. We do not downplay security impact or bury vulnerabilities with silent fixes, because doing so serves users poorly. Instead, we provide users and administrators with the information they need to accurately assess risk. The security team publishes how it handles security issues, discloses all vulnerabilities fixed in Chrome and its dependencies—whether discovered internally or externally–and, whenever possible, lists every fixed security issue in its release notes.

Beyond vulnerabilities, we share quarterly summaries about what we're doing with the public and engage with users via an external discussion mailing list (*security-dev@chromium.org*) so that anyone can send us ideas, questions, or participate in ongoing discussions. We actively encourage individuals to share their work at conferences or security gatherings or via their social networks. We also engage with the larger security community via Chrome's Vulnerability Reward Program and security conference sponsorship. Chrome is more secure thanks to the contributions of many people who don't identify as being part of the team, and we do our best to acknowledge and reward those contributions by ensuring proper attribution and paying out monetary rewards.

Across Google, we organized an annual offsite meeting to connect the Chrome security team with the central security team and other embedded teams (for example, Android's security team). We also encourage the security enthusiasts across Google to do 20% of their work on Chrome (and vice versa), or find opportunities to collaborate with academic researchers on Chromium projects.

Working in an open environment allows the team to share its work, accomplishments, and ideas, and to get feedback or pursue collaborations beyond the confines of Google. All of this contributes to advancing the common understanding of browser and web security. The team's multiyear effort to increase HTTPS adoption, described in Chapter 7, is one example of how communicating changes and engaging with a larger community can lead to ecosystem change.

Conclusion

The team working on Chrome identified security as a core principle in the early stages of the project, and scaled its investment and strategy as the team and user base grew. The Chrome security team was formally established just a year after the browser's launch, and over time its roles and responsibilities became more clearly

defined. The team articulated a mission and a set of core security principles, and established key focus areas for its work.

Promoting security as a team responsibility, embracing transparency, and engaging with communities outside of Chrome helped to create and advance a security-centric culture. Aiming to innovate at speed led to a dynamic product and the ability to respond with agility to a changing landscape. Designing for defense in depth helped to protect users from one-off bugs and novel attacks. Considering the human aspects of security, from the end-user experience to the hiring process, helped the team expand its understanding of security and address more complex challenges. A willingness to meet challenges head-on and learn from mistakes enabled the team to work toward making the default path the secure one as well.