An Empirical Study of Vulnerability Rewards Programs

Matthew Finifter, Devdatta Akhawe, and David Wagner
University of California, Berkeley
{finifter, devdatta, daw}@cs.berkeley.edu

Abstract

We perform an empirical study to better understand two well-known vulnerability rewards programs, or VRPs, which software vendors use to encourage community participation in finding and responsibly disclosing software vulnerabilities. The Chrome VRP has cost approximately $580,000 over 3 years and has resulted in 501 bounties paid for the identification of security vulnerabilities. The Firefox VRP has cost approximately $570,000 over the last 3 years and has yielded 190 bounties. 28% of Chrome's patched vulnerabilities appearing in security advisories over this period, and 24% of Firefox's, are the result of VRP contributions. Both programs appear economically efficient, comparing favorably to the cost of hiring full-time security researchers. The Chrome VRP features low expected payouts accompanied by high potential payouts, while the Firefox VRP features fixed payouts. Finding vulnerabilities for VRPs typically does not yield a salary comparable to a full-time job; the common case for recipients of rewards in either program is that they have received only one reward. Firefox has far more critical-severity vulnerabilities than Chrome, which we believe is attributable to an architectural difference between the two browsers.

1 Introduction

Some software vendors pay security researchers for the responsible disclosure of a security vulnerability. Programs implementing the rules for this exchange are known as vulnerability rewards programs (VRPs) or bug bounty programs. The last couple of years have seen an upsurge of interest in VRPs, with some vendors expanding their existing programs [1, 19], others introducing new programs [3, 34, 38], and some companies offering to act as an intermediary between security researchers and vendors offering VRPs [53].

VRPs offer a number of potential attractions to software vendors. Offering adequate incentives entices security researchers to look for vulnerabilities, and this increased attention improves the likelihood of finding latent vulnerabilities.[1] Second, coordinating with security researchers allows vendors to more effectively manage vulnerability disclosures, reducing the likelihood of unexpected and costly zero-day disclosures. Monetary rewards provide an incentive for security researchers not to sell their research results to malicious actors in the underground economy or the gray world of vulnerability markets. Third, VRPs may make it more difficult for black hats to find vulnerabilities to exploit. Patching vulnerabilities found through a VRP increases the difficulty and therefore cost for malicious actors to find zero-days because the pool of latent vulnerabilities has been diminished. Additionally, experience gained from VRPs (and exploit bounties [23, 28]) can yield improvements to mitigation techniques and help identify other related vulnerabilities and sources of bugs. Finally, VRPs often engender goodwill amongst the community of security researchers. Taken together, VRPs provide an attractive tool for increasing product security and protecting customers.

Despite their potential benefits, there is an active debate over the value and effectiveness of VRPs. A number of vendors, notably Microsoft,[2] Adobe, and Oracle, do not maintain a VRP, with Microsoft arguing that VRPs do not represent the best return on investment on a per-bug basis [26]. Further, it is also not clear if the bounties awarded are a sufficient attraction for security researchers motivated by money—underground economy prices for vulnerabilities are far higher than those offered by VRPs [20, 37].

Given the emergence of VRPs as a component of the secure development lifecycle and the debate over the efficacy of such programs, we use available data to better understand existing VRPs. We focus on the Google Chrome and Mozilla Firefox web browsers, both of which are widely considered to have mature VRPs, as case studies. We analyze these VRPs along several dimensions with the intention of better understanding the characteristics, metrics, and trajectory of a VRP.

We make the following contributions:

- We collect and analyze data on vulnerability rewards over the last 3 years for the Google Chrome VRP and the Mozilla Firefox VRP (Section 3).
- We assess the state of these VRPs along several dimensions, including costs, benefits, popularity, and

[1] For instance, Linus's Law suggests "Given enough eyeballs, all bugs are shallow." [48]

[2] On June 19, 2013, during final preparation of this manuscript, Microsoft announced a month-long VRP for the IE11 developer preview [54].

efficacy (Section 4), finding that these VRPs appear both effective and cost-effective.

- We make concrete recommendations for software vendors aiming to start or evolve their own VRP (Section 5.2).

- We generate hypotheses, which identify opportunities for future research on VRPs and secure software development.

2 Background

A *secure software development lifecycle (SDLC)* aims to address software security throughout the entire software development process, from before specifications are developed to long after software has been released [15]. A *vulnerability remediation strategy* is any systematic approach whose goal is to reduce the number of software vulnerabilities [57]. Vulnerability remediation strategies are one important part of an SDLC, complemented by things like incident response [32], operational security considerations [46], and defense in depth [16].

Potential vulnerability remediation strategies include:

- **Code reviews.** These can range from informal, as-needed requests for code review to systematized, formal processes for mandatory code inspection. Typically, SDLCs also include an extra security review for security critical features.

- **Penetration testing.** Software vendors may perform in-house penetration testing or may hire external companies who specialize in this service. Penetration testing ranges from manual to automated.

- **Use of dynamic and static analysis tools.** Specialized tools exist for catching a wide range of flaws, e.g., memory safety vulnerabilities, configuration errors, and concurrency bugs.

- **Vulnerability rewards programs.** The focus of our study, VRPs have recently received increased attention from the security community.

How such strategies are systematized and realized varies widely between software vendors. One company might require mandatory code reviews before code check-in, while another might hire outside penetration testing experts a month before product release. Vendors often combine or innovate on existing strategies.

Vulnerability rewards programs (VRPs) appear to be emerging as a viable vulnerability remediation strategy. Many companies have them, and their popularity continues to grow [6, 9]. But VRPs have gone largely unstudied. For a company considering the use of a VRP in their SDLC, guidance is limited.

By studying mature, high-profile VRPs, we aim to provide guidance on the development of new VRPs and the evolution and maturation of existing VRPs. Vendors looking to grow their VRPs can benefit from an improved understanding of those VRPs we study.

Toward this end, we measure, characterize, and discuss the Google Chrome and Mozilla Firefox VRPs. We choose these VRPs in particular because browsers are a popular target for malicious actors today. Their ubiquitous nature and their massive, complex codebase with significant legacy components make them especially vulnerable. Complex, high-performance components with a large attack surface such as JavaScript JITs also provide an alluring target for malicious actors. For the same reasons, they are also widely studied by security researchers; they therefore provide a large sample size for our study. In addition, browser vendors were among the first to offer rewards for vulnerabilities: Mozilla's VRP started in 2004 and Google introduced the Chrome VRP in 2010, before the security community at large adopted VRPs as a vulnerability remediation strategy.

2.1 Goals

We intend to improve our understanding of the following characteristics of a mature VRP: (1) Expected cost, (2) expected benefits, (3) incentive levels effective for encouraging and sustaining community participation, and (4) volume of VRP activity (e.g., number of patches coming out of VRP reports).

We do so by studying available data coming out of two exemplars of well-known, mature VRPs, that of Google Chrome and Mozilla Firefox. Understanding these VRPs will allow these vendors to evaluate and improve their programs, and it will suggest targets for other vendors to strive toward with their VRPs. At minimum, we hope to arrive at a better understanding of the current state of VRPs and how they have evolved. At best, we aim to make concrete suggestions for the development and improvement of VRPs.

2.2 Google Chrome VRP

The Google Chrome VRP[3] is widely considered an exemplar of a mature, successful VRP. When first introduced in January 2010, the Google Chrome VRP offered researchers rewards ranging from $500 for high- and critical-severity bugs, with a special $1337 reward for particularly critical or clever bugs. Over time, the typical payout increased to a $1000 minimum with a maximum payout of $3133.7 for high-impact vulnerabilities. Additionally, the Chrome team, has provided rewards of up to $31,336 for exceptional vulnerability reports [21].

[3]The program is officially the Chromium VRP with prizes sponsored by Google. We refer to it as the Google Chrome VRP for ease of exposition.

Google also sponsors a separate, semi-regular exploit bounty called the "pwnium" competition [23]. This program focuses on full exploits; a mere vulnerability is not enough. In return, it awards higher bounties (as high as $150,000) for these exploits [8]. Reliable exploits for modern browsers typically involve multiple vulnerabilities and significant engineering effort. For example, the two winning entries in a recent "pwnium" contest required six and ten vulnerabilities in addition to "impressive" engineering in order to achieve a successful exploit [7,45]. Our focus is on programs that provide bounties for vulnerabilities; we do not consider exploit bounties in this work.

The severity of a vulnerability plays a key role in deciding reward amounts. Google Chrome uses a clear guideline for deciding severity [12]. In short, a critical vulnerability allows an attacker to run arbitrary native code on the user's machine; for instance, web-accessible memory corruption vulnerabilities that appear in the Chrome kernel[4] are typically critical severity. A high-severity vulnerability is one that allows an attacker to bypass the same-origin policy, e.g., via a Universal XSS vulnerability (which enables an attacker to mount an XSS attack on any web site) or a memory corruption error in the sandbox. A vulnerability is of medium severity if achieving a high/critical status requires user interaction, or if the vulnerability only allows limited disclosure of information. Finally, a low-severity vulnerability refers to all the remaining security vulnerabilities that do not give the attacker control over critical browser features. Medium-severity vulnerabilities typically receive rewards of $500, and low-severity vulnerabilities typically do not receive rewards.

2.3 Mozilla Firefox VRP

Mozilla's VRP is, to the best of our knowledge, one of the oldest VRPs in the industry. It was first started in 2004 and based on a similar project at Netscape in 1995 [41]. The Mozilla VRP initially awarded researchers $500 for high- or critical-severity security bugs. Starting July 1, 2010 Mozilla expanded its program to award all high/critical vulnerabilities $3000 [1].

Mozilla's security ratings are similar to that of Chrome. Critical vulnerabilities allow arbitrary code execution on the user's computer. Vulnerabilities that allow an attacker to bypass the same-origin policy or access confidential information on the user's computer are high severity. Due to the absence of privilege separation in the Firefox browser, *all* memory corruption vulnerabilities are critical, regardless of the component affected. Mozilla is currently investigating a privilege-separated design for Firefox [17,36,39].

Mozilla's VRP also qualitatively differs from the Google program. First, Mozilla awards bounties even if the researcher publicly discusses the vulnerability instead of reporting it to Mozilla.[5] Second, Mozilla also explicitly awards vulnerabilities discovered in "nightly" (or "trunk") versions of Firefox. In contrast, Google discourages researchers from using "canary" builds and only awards bounties in canary builds if internal testing would miss those bugs [55].

3 Methodology

For both browsers, we collect all bugs for which rewards were issued through the browser vendor's VRP. To evaluate the impact of the VRP as a component of the SDLC, we also collected all security bugs affecting stable releases. We chose to look only at bugs affecting stable releases to ignore the impact of transient bugs and regressions caught by internal testing.

For each bug in the above two datasets, we gathered the following details: (1) severity of the bug, (2) reward amount, (3) reporter name, (4) report date. For bugs affecting stable releases, we also aimed to gather the date a release patching the bug became available for download. As we discuss below, we were able to gather this data for only a subset of all bugs.

For all bugs, we mark a bug as internally or externally reported via a simple heuristic: if a reward was issued, the reporter was external, and otherwise the reporter was internal. Because low and medium severity vulnerabilities usually do not receive bounties, we only look at critical/high vulnerabilities when comparing internal and external bug reports. While all high/critical vulnerabilities are eligible for an award, a researcher can still refuse an award, in which case, our heuristic falsely marks the bug "internal." We are aware of a handful of such instances, but there are not enough of these in our dataset to affect our analysis.

We are also aware of some researchers who transitioned from external to internal over the course of our study period. Because our heuristic operates on a per-bug basis (as opposed to marking each person as internal or external), the same person may be (intentionally) considered internal for one bug and external for another.

In this section, we present how we gathered this dataset for Chrome and Firefox. We first discuss how we identify the list of bugs affecting stable releases and bugs awarded bounties, followed by a discussion on how we identified, for each bug, other details such as severity. Finally, we discuss threats to the validity of our measurement study.

3.1 Gathering the Google Chrome dataset

We gathered data from the official Chromium bug tracker [13] after confirming with Google employees that

[4]Chrome follows a privilege-separated design [4]. The Chrome kernel refers to the privileged component.

[5]But Mozilla reports that this was a rare occurrence over the period of time we consider, possibly because the VRP became widely known [56].

the bug tracker contained up-to-date, authoritative data on rewards issued through their VRP. We search the bug tracker for bugs marked with the special "Reward" label to collect bugs identified through the VRP. Next, we searched the bug tracker for bugs marked with the special "Security-Impact: Stable" to collect bugs affecting stable releases. Next, we remove the special Pwnium [23] rewards from all datasets because Pwnium rewards *exploits* instead of vulnerabilities as in the regular VRP. This provides us with 501 bugs identified through the VRP and 1347 bugs affecting stable releases.

The Chromium Bug tracker provides a convenient interface to export detailed bug metadata, including severity, reporter, and report date, into a CSV file, which we use to appropriately populate our dataset. We identify the reward amounts using the "Reward" label.

Unfortunately, the Chromium bug tracker does not include the release date of bug fixes. Instead, we gather this data from the Google Chromium release blog [27]. For each stable release of the Chromium browser, Google releases a blog post listing security bugs fixed in a release. For the subset of bugs mentioned in these release notes, we extract the release date of the stable version of Chrome that patches the bug.

3.2 Gathering the Mozilla Firefox dataset

Similar to Google Chrome, we searched Bugzilla, the Firefox bug tracker, for an attachment used to tag a bug bounty.[6] We identified 190 bugs via this search.

Unlike the Chrome bug tracker, Bugzilla does not provide a convenient label to identify bugs affecting stable releases. Instead, Mozilla releases Mozilla Foundation Security Advisories (MFSA) with every stable release of Mozilla Firefox [40]. We scraped these advisories for a list of bugs affecting stable releases. We also use the MFSAs to identify the release date of a patched, stable version of Firefox. We gathered 613 unique bugs from the MFSA advisories dating back to March 22, 2010 (Firefox 3.6).

Similar to the Chromium Bug tracker, the Bugzilla website provides a convenient interface to export detailed bug data into a CSV file for further analysis. We used Bugzilla to collect, for each bug above, the bug reporter, the severity rating, and the date reported. The security severity rating for a bug is part of the Bugzilla keywords field and not Bugzilla's severity field. We do not separately collect the amount paid because, as previously discussed, Mozilla's expanded bounty program awards $3,000 for all critical/high vulnerabilities.

[6]The existence of this attachment is not always visible to the public. We acknowledge the support of Mozilla contributor Dan Veditz for his help gathering this data.

Severity	Chrome		Firefox	
	Stable	Bounty	Stable	Bounty
Low	226	1	16	1
Medium	288	72	66	9
High	793	395	79	38
Critical	32	20	393	142
Unknown	8	13	59	0
Total	1347	501	613	190

Table 1: Number of observations in our dataset.

3.3 Dataset

Table 1 presents information about the final dataset we used for our analysis. We have made our dataset available online for independent analysis [33].

3.4 Threats to validity

In this section, we document potential threats to validity so readers can better understand and take into account the sources of error and bias in our study.

It is possible that our datasets are incomplete, i.e., there exist patched vulnerabilities that do not appear in our data. For example, for both Chrome and Firefox, we rely heavily on the keyword/label metadata to identify bugs; since this labeling is a manual process, it is possible that we are missing bugs. To gather the list of bugs affecting stable releases, we use the bug tracker for Chrome but use security advisories for Mozilla, which could be incomplete. Given the large number of vulnerabilities we do have in our datasets, we expect that a small number of missing observations would not materially influence the conclusions we draw.

We treat all rewards in the Firefox VRP as $3,000 despite knowing that 8% of the rewards were for less than this amount [56]. Data on which rewards were for less money and what those amounts were is not publicly available. Any results we present regarding amounts paid out for Firefox vulnerabilities may therefore have as much as 8% error, though we expect a far lower error, if any. We do not believe this affects the conclusions of our analysis.

Parts of our analysis also compare Firefox and Chrome VRPs in terms of number of bugs found, which assumes that finding security vulnerabilities in these browsers requires comparable skills and resources. It could be the case that it is just easier to find bugs in one over the other, or one browser has a lower barrier to entry for vulnerability researchers. For example, the popular Address Sanitizer tool worked only on Google Chrome until Mozilla developers tweaked their build infrastructure to enable support for the same [31]. Another confound is the possibility that researchers target a browser based on personal factors beyond VRPs. For example, researchers could look for vulnerabilities only in the browser they personally use.

Assigning bug severity is a manual process. While the severity assignment guidelines for both browsers are similar, it is possible that vendors diverge in their actual severity assignment practices. As a result, the bug severities could be incomparable across the two browsers.

We study only two VRPs; our results do not necessarily generalize to any other VRPs. We caution the reader to avoid generalizing to other VRPs, but instead take our results as case studies of two mature, well-known VRPs.

4 Results

We study VRPs from the perspectives of three interested parties: the software vendor, the independent security researcher, and the security researcher employed full-time by the software vendor.

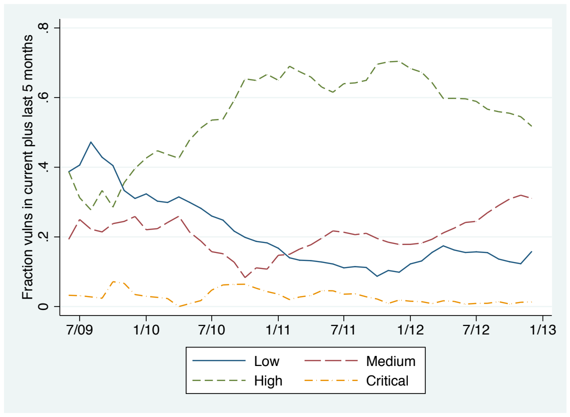
4.1 Software vendor

We model the software vendor's goal as follows: to increase product security as much as possible while spending as little money as possible. There are many potential strategies for working toward this goal, but in this paper we consider the strategy of launching a VRP. We present data on costs and benefits for two VRPs, and generate hypotheses from this data. The software vendor's motivation can also include publicity and engaging the security research community. We do not measure the impact of VRPs on these.

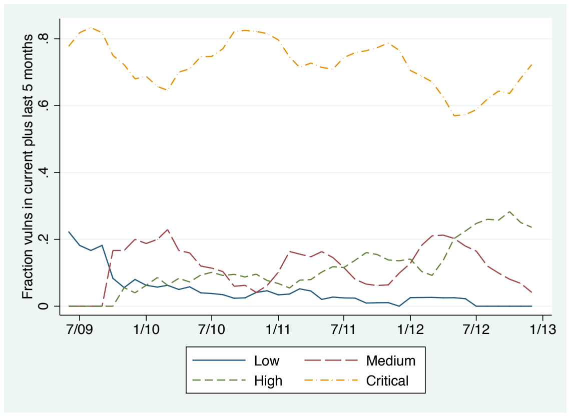
4.1.1 Number of vulnerabilities

The intended benefit of a VRP is to improve product security. A reduction in the number of latent vulnerabilities is one way of improving product security. We find that the Chrome VRP uncovers about 2.6 times as many vulnerabilities as that of Firefox (501 vs. 190), despite the fact that Chrome's total number of security vulnerabilities in our dataset is only 2.2 times that of Firefox (Table 1). 27.5% of bugs affecting Chrome releases originate from VRP contributions (371 of 1347), and 24.1% of bugs affecting Firefox releases (148 of 613) result from VRP contributions.

Discussion Both VRPs yield a significant fraction of the total number of security advisories, which is a clear benefit. Chrome is seeing approximately 1.14 times the benefit of Firefox by our metric of fraction of advisories resulting from VRP contributions. We only study bugs affecting stable releases in this metric and caution the reader from assuming that VRPs are competitive with internal researchers. For both browsers, internal researchers find far more bugs during the months of testing that precede a typical browser release. For example, from January to May 2013, across all release channels, Google researchers found 140 high or critical vulnerabilities in Chrome, while the Chrome VRP only found 40 vulnerabilities in the same time period.



(a) Chrome



(b) Firefox

Figure 1: Moving average over the current plus 5 previous months of the percentage of vulnerabilities at each severity level (low is blue solid line, medium is red long-dashed line, high is green short-dashed line, and critical is yellow dash-dotted line). In this and subsequent line graphs, the data are aggregated by month to improve graph readability, and the x-axis represents the open date of the bug.

4.1.2 Vulnerability severity

Another measure of improvement to product security is change in vulnerability severity over time. It is a good sign, for example, if the percentage of critical-severity vulnerabilities has decreased over time.

Table 1 lists the total number of vulnerabilities by severity for Firefox and Chrome. Figure 1 plots the fraction of vulnerabilities at each severity level over the current plus 5 previous months.

Discussion Firefox has a much higher ratio of critical vulnerabilities to high vulnerabilities than Chrome. We expect that many of Firefox's critical vulnerabilities would instead be high severity if, like Chrome, it also had a privilege-separated architecture. The lack of such an architecture means that any memory corruption vulnerability in Firefox is a critical vulnerability. We therefore hypothesize that:

Hypothesis 1 *This architectural difference between Chrome and Firefox—that the former is privilege-separated and the latter is not—is the most influential factor in causing such a large difference in vulnerability severity classification.*

The fraction of critical severity bugs has remained relatively constant for Chrome. We also notice the start of a trend in Chrome—the fraction of high severity vulnerabilities is declining and the fraction of medium severity vulnerabilities is increasing.

Chrome's privilege-separated architecture means that a critical vulnerability indicates malicious code execution in the privileged process. We see that there continue to be new bugs resulting in code execution in the privileged process. Further investigation into these bugs can help understand how and why they continue to surface.

Low-severity vulnerabilities in Google Chrome make up a significant fraction of all vulnerabilities reported. In contrast, the fraction of low- and medium-severity vulnerabilities in Firefox remains negligible.

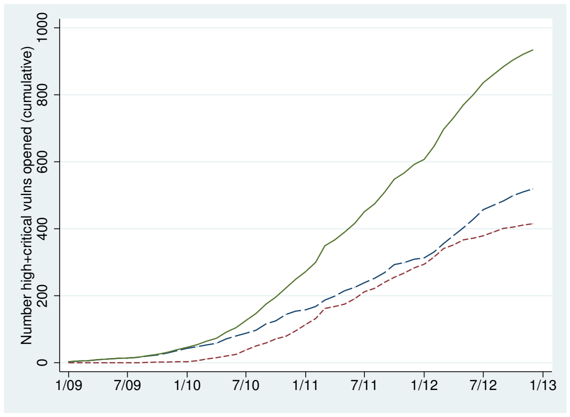
Note that our dataset does not allow us to attribute any change in vulnerability severity over time to the use or success of a VRP. However, severity over time is a metric worth tracking for a software vendor because it can indicate trends in the overall efforts to improve product security, of which a VRP may be one component.

4.1.3 Community engagement

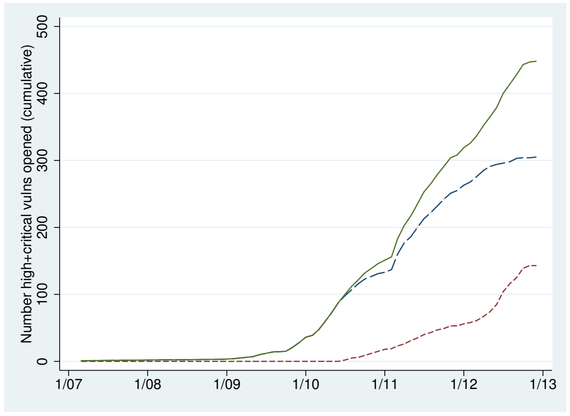
One advantage of VRPs is engagement with the broader security community. We studied this engagement along two axes: (1) the contribution of internal and external researchers towards identifying security vulnerabilities, and (2) the number of external participants in each VRP.

Figure 2 depicts the cumulative number of high- and critical-severity vulnerabilities patched and Figure 3 depicts the same, but for only critical vulnerabilities. Table 2 shows the distribution of the total number of vulnerabilities reported by each external participant in each of the two VRPs. Although a few external participants submit many bugs, there is a clear long tail of participants in both VRPs. Table 3 shows the same distribution, but for internal (i.e., employee) reporters of vulnerabilities.

Discussion For both browsers, internal contributions for high- and critical-severity vulnerabilities have consistently yielded the majority of patches. The number of external contributions to Chrome has nearly caught up with the number of internal contributions (i.e., around 4/11 and 3/12, in Figure 2a) at various times, and as of the end of our study, these two quantities are comparable. Considering only critical-severity vulnerabilities, external contributions have exceeded internal contributions as of



(a) Chrome



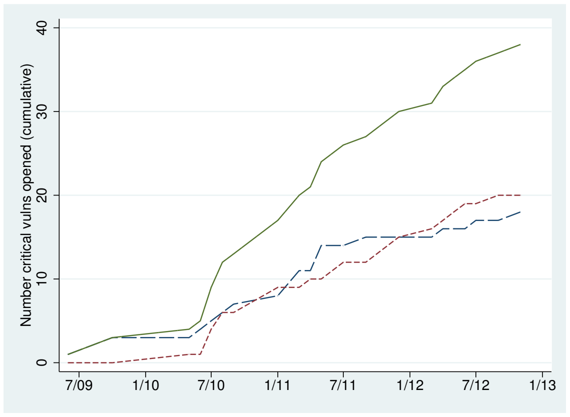
(b) Firefox

Figure 2: Number of high- plus critical-severity vulnerabilities reported over time, discovered internally (blue long-dashed line), externally (red short-dashed line), and total (green solid line).

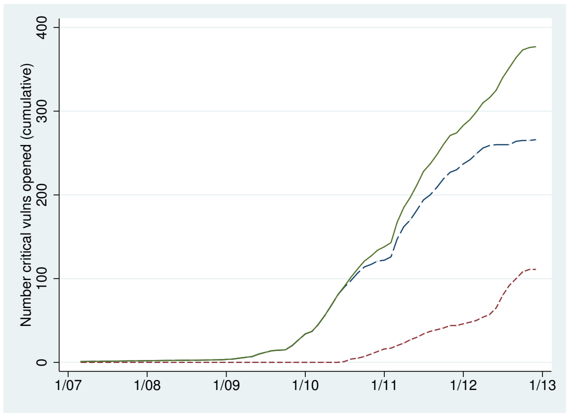
the end of our study. For Firefox, on the other hand, the number of external contributions has consistently been far lower than the number of internal contributions.

We observe an increase in the rate of external contributions to Chrome starting around July 2010, six months after the inception of the VRP. As seen in Figure 3a, this is more pronounced when considering only critical-severity vulnerabilities. We conjecture that this change corresponds to increased publicity for the Chrome VRP after Google increased reward amounts [19].

Linus's Law, defined by Eric Raymond as "Given enough eyes, all bugs are shallow," suggests that it is in the interests of the software vendor to encourage more people to participate in the search for bugs. The distributions of bugs found by external participants indicate that both VRPs have been successful in encouraging broad community participation. The existence of a long tail of contributors holds for internal contributors as well as external contributors.



(a) Chrome



(b) Firefox

Figure 3: Number of critical-severity vulnerabilities reported over time, discovered internally (blue long-dashed line), externally (red short-dashed line), and total (green solid line).

4.1.4 Diversity

There is the potential benefit that the wide variety of external participants may find *different* types of vulnerabilities than internal members of the security team. A few pieces of anecdotal evidence support this. Chrome has awarded bounty amounts that include $1,337, $2,337, $3,133.7, and $7,331 for bugs that they considered clever or novel [21], and our dataset contains 31 such awards. Additionally, one of PinkiePie's Pwnium exploits led to a full review of the Chrome kernel file API, which resulted in the discovery of several additional vulnerabilities [21, 51]. The Chrome security team missed all these issues until PinkiePie discovered and exploited one such issue [14]. We therefore hypothesize that:

Hypothesis 2 *An increase in the number of researchers looking for vulnerabilities yields an increase in the diversity of vulnerabilities discovered.*

# Bugs	Freq.
1	45
3	2
4	1
6	1
10	1
12	2
13	3
16	1
17	1
22	1
24	1
27	1
35	1
48	1
92	1
Total	63

(a) Chrome

# Bugs	Freq.
1	46
2	9
3	4
5	1
6	1
9	1
10	1
12	1
14	1
47	1
Total	66

(b) Firefox

Table 2: Frequency distribution of number of high- or critical-severity vulnerabilities found by external contributors.

# Bugs	Freq.
1	67
2	10
3	10
4	2
5	2
14	1
20	2
67	1
263	1
Total	96

(a) Chrome

# Bugs	Freq.
1	43
2	10
3	7
4	3
5	2
6	2
7	1
12	1
13	1
15	1
17	2
18	1
21	1
23	1
44	1
Total	77

(b) Firefox

Table 3: Frequency distribution of number of high- or critical-severity bugs found by internal contributors.

4.1.5 Cost of rewards

Though the number of bounties suggests that VRPs provide a number of benefits, a thorough analysis necessarily includes an analysis of the costs of these programs. In this section, we examine whether VRPs provide a *cost-effective* mechanism for software vendors. We analyze one ongoing cost of the VRP: the amount of money paid to researchers as rewards for responsible disclosure. Running a VRP has additional overhead costs that our dataset does not provide any insight into.

Figure 4 displays the total cost of paying out rewards for vulnerabilities affecting stable releases. We find that over the course of three years, the costs for Chrome and Firefox are similar: approximately $400,000.

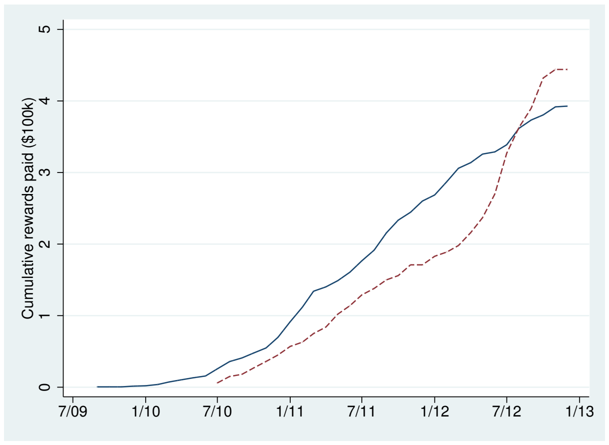


Figure 4: Cumulative rewards paid out for Chrome (blue solid line) and Firefox (red dashed line), excluding rewards for vulnerabilities not affecting stable versions.

Rewards for Development Releases Both Firefox and Chrome issue rewards for vulnerabilities that do not affect stable release versions, increasing the *total* cost of the VRP beyond the cost of rewarding vulnerabilities affecting stable releases. One potential drawback of such rewards is that the VRPs awards transient bugs that may never make their way into a user-facing build in the first place. On the other hand, such rewards could catch bugs earlier in the development cycle, reducing the likelihood of expensive out-of-cycle releases.

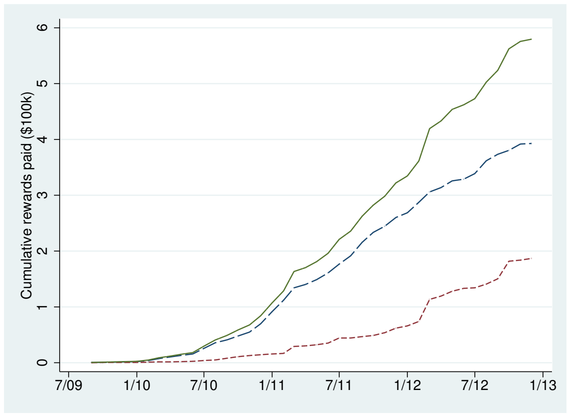
Figure 5 shows the cumulative rewards issued by each of the two VRPs for vulnerabilities affecting stable releases, vulnerabilities not affecting stable releases, and the sum of the two. We observe that the Chrome VRP has paid out $186,839, 32% of its total cost of $579,605 over the study period for vulnerabilities not affecting a stable release. The Firefox VRP has paid out $126,000, 22% of its total cost of $570,000, over the study period for such vulnerabilities.

Discussion The total cost of each of the two VRPs is remarkably similar. Both spend a significant fraction of the total cost on vulnerabilities not present in stable release versions.

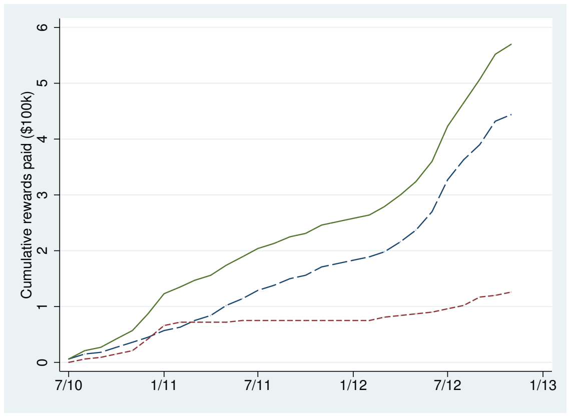
4.1.6 Average daily cost

Figure 6 plots the average daily cost to date of each VRP over time. We see that Chrome's VRP has cost $485 per day on average, and that of Firefox has cost $658 per day.

Discussion If we consider that an average North American developer on a browser security team (i.e., that of Chrome or Firefox) would cost the vendor around $500 per day (assuming a $100,000 salary with a 50% overhead), we see that the cost of either of these VRPs is comparable to the cost of just one member of the browser security team. On the other hand, the *benefit* of a VRP



(a) Chrome



(b) Firefox

Figure 5: Cumulative rewards paid out for vulnerabilities affecting a stable release (blue long-dashed line), vulnerabilities not affecting any stable release (red short-dashed line), and the sum of the two (green solid line).

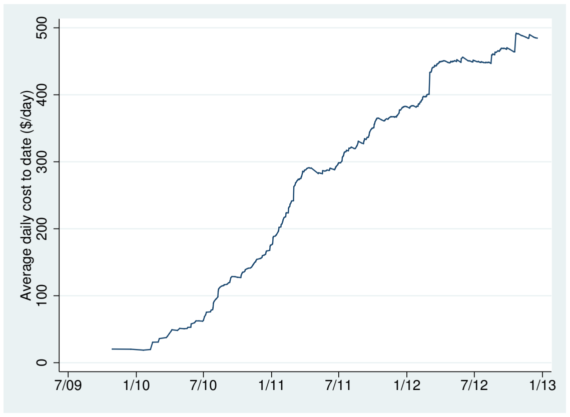
far outweighs that of a single security researcher because each of these VRPs finds many more vulnerabilities than any one researcher is likely to be able to find. For bugs affecting stable releases, the Chrome VRP has paid 371 bounties, and the most prolific internal security researcher has found 263 vulnerabilities. For Firefox, these numbers are 148 and 48, respectively. Based on this simple cost/benefit analysis, we hypothesize that:

Hypothesis 3 *A VRP can be a cost-effective mechanism for finding security vulnerabilities.*

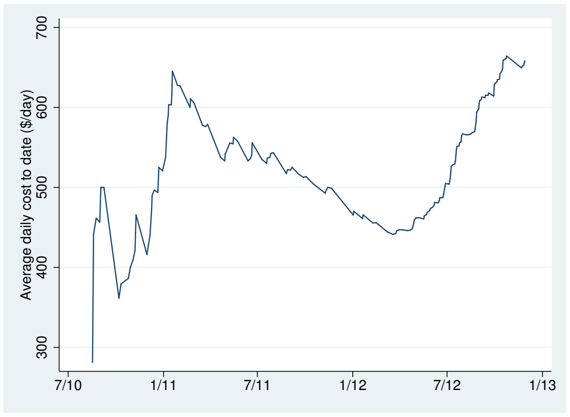
4.2 External security researcher

We model the goal of an external security researcher as follows: to make as much money as possible in as little time as possible.[7] The researcher can contribute to

[7]Naturally, this does not reflect the reality of every security researcher's goal.



(a) Chrome



(b) Firefox

Figure 6: Average daily cost to date since first reward.

any VRP he chooses, each of which pays out according to some rewards distribution. The researcher has some perception of security of each product, which reflects the expected amount of time the researcher will have to spend to find a vulnerability.

A rational strategy for the security researcher is to look for products perceived to be insecure that also happen to pay large bounties. This implies that a product with a higher perceived security must pay relatively higher bounties to encourage researchers to look for vulnerabilities in it as opposed to in a different product that is perceived to be less secure. We therefore hypothesize that:

Hypothesis 4 *In an efficient market with many VRPs and fluid reward structures, larger rewards reflect a higher level of perceived security by the population of researchers who contribute to VRPs.*

4.2.1 Reward amounts

Our dataset provides insight into the distributions of rewards for two products. Firefox offers a standard reward of $3,000 for all vulnerabilities. In contrast, the Chrome

Amount ($)	Frequency (%)
500	24.75
1,000	60.08
1,337	3.59
1,500	2.99
2,000	2.99
2,337	0.60
2,500	0.60
3,000	0.20
3,133	1.80
3,500	0.20
4,000	0.20
4,500	0.20
5,000	0.20
7,331	0.20
10,000	1.40

Table 4: Percentage of rewards given for each dollar amount in Chrome VRP.

VRP's incentive structure provides different reward levels based on a number of subjective factors like difficulty of exploit, presence of a test case, novelty, and impact, all of which is at the discretion of Google developers.

Table 4 depicts the reward amounts paid to external researchers by the Chrome VRP. The majority of the rewards are for only $500 or $1,000. Large rewards, such as $10,000 rewards, are infrequent.

Discussion We hypothesize that high maximum rewards entice researchers to participate, but low ($500 or $1,000) rewards are typical, and the total cost remains low. The median (mean) payout for Chrome bug bounty is $1,000 ($1,156.9), suggesting that a successful VRP can be inexpensive with a low expected individual payout. Much like the lottery, a large maximum payout ($30,000 for Chrome), despite a small expected return (or even negative, as is the case of anyone who searches for bugs but never successfully finds any) appears to suffice in attracting enough participants. Bhattacharyya and Garrett [5] explain this phenomenon as follows:

> Lotteries are instruments with negative expected returns. So when people buy lottery tickets, they are trading off negative expected returns for skewness. Thus, if a lottery game has a larger prize amount, then a buyer will be willing to accept a lower chance of winning that prize.

4.2.2 VRPs as employment

Our dataset also allows limited insight into how much money independent security researchers make. Table 5a displays the total amounts of money earned by each external contributor to the Chrome VRP. Only three external contributors (out of eighty two) have earned over $80,000 over the lifetime of the VRP, and an additional five have earned over $20,000.

$ earned	Freq.
500	26
1,000	25
1,337	6
1,500	2
2,000	1
3,000	2
3,133	1
3,500	2
4,000	1
5,000	1
7,500	1
11,000	1
11,500	1
11,837	1
15,000	1
17,133	1
18,337	1
20,633	1
24,133	1
28,500	1
28,633	1
37,470	1
80,679	1
85,992	1
105,103	1
Total	82

(a) Chrome

$ earned	Freq.
3,000	46
6,000	12
9,000	4
12,000	1
15,000	1
21,000	1
27,000	1
30,000	1
36,000	1
42,000	1
141,000	1
Total	70

(b) Firefox

Table 5: Frequency distributions of total amounts earned by external VRP contributors.

In contrast to Google Chrome, we see in Table 5b that a single Firefox contributor has earned $141,000 ($47,000 per year) since the beginning of our study period. Ten of this individual's rewards, representing $30,000, were for vulnerabilities that did not impact a stable release. Six contributors have earned more than $20,000 via the Mozilla VRP.

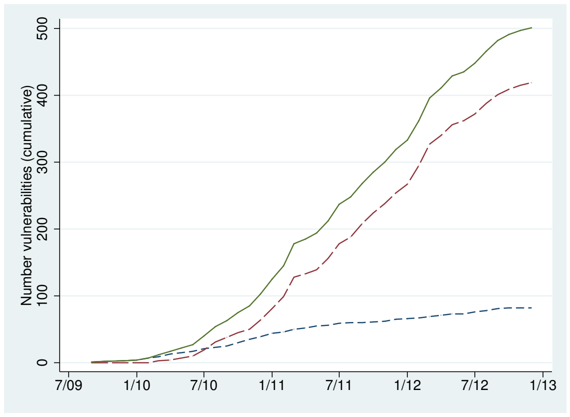
Discussion Based on the data from 2 VRPs, we hypothesize that:

Hypothesis 5 *Contributing to a single VRP is, in general, not a viable full-time job, though contributing to multiple VRPs may be, especially for unusually successful vulnerability researchers.*

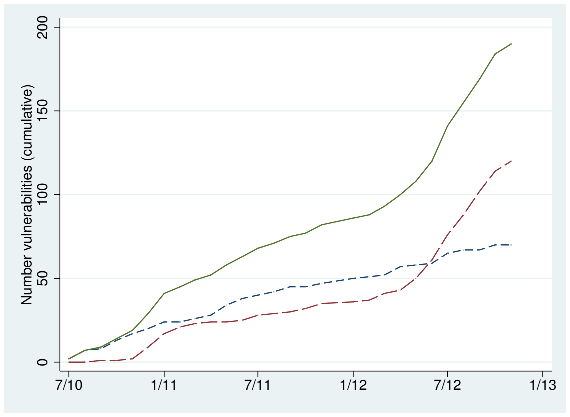
4.2.3 Repeat contributors

Figure 7 shows the cumulative number of vulnerabilities patched due to reports from first-time VRP participants and repeat participants. For both programs, first-time participant rewards are steadily increasing, and repeat participant rewards are increasing even more quickly.

Discussion Both VRP incentive structures are evidently sufficient for both attracting new participants and continuing to entice existing participants, though we do note differences between Chrome and Firefox. Until recently, repeat participants in Firefox's VRP represented a relatively small fraction of the number of awards issued.



(a) Chrome



(b) Firefox

Figure 7: Cumulative number of vulnerabilities rewarded, as reported by (1) first-time VRP contributors (blue short-dashed line), (2) repeat contributors (red long-dashed line), and (3) all contributors (green solid line).

Chrome, on the other hand, has seen the majority of its reports come from repeat participants for nearly the whole lifetime of its VRP.

4.3 Internal security researcher

An internal security researcher is a full-time employee of a software vendor who is paid a salary to find as many vulnerabilities as possible. Google hired at least three researchers who first came to light via the Chrome VRP [21] and Mozilla hired at least three researchers as well [56].

Discussion A software vendor may choose to hire an unusually successful independent security researcher. The researcher's past performance indicates how many vulnerabilities the vendor can expect them to find, and the vendor may prefer to pay a fixed salary instead of a per-vulnerability reward. The researcher may also prefer this; the researcher trades a potentially higher amount of cash for less compensation, but more benefits and job security.

Severity	Median, Chrome	Std. dev. Chrome	Median, Firefox	Std. dev., Firefox
Low	58.5	110.6	114	256.1
Medium	45.5	78.9	106	157.6
High	28.0	35.3	62.5	85.7
Critical	20.0	26.6	76	116.5

Table 6: Median and standard deviation of number of days between vulnerability report and release that patches the vulnerability, for each severity level.

Accordingly, we hypothesize that:

Hypothesis 6 *Successful independent security researchers bubble to the top, where a full-time job awaits them.*

4.4 Other factors

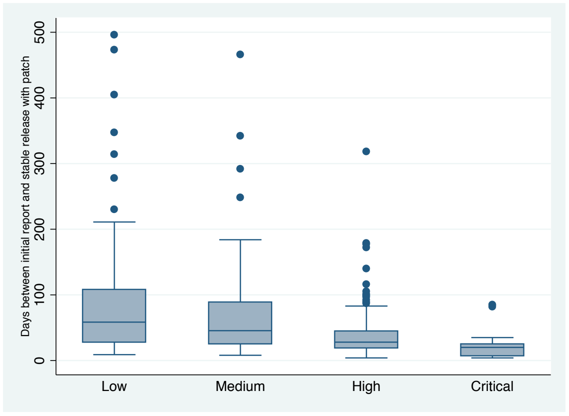
Our dataset provides an additional opportunity to better understand the state of the SDLC (software development lifecycle) at Mozilla and Google. In particular, we analyze (1) the elapsed time between a vulnerability report and the release of a patched browser version that fixes the vulnerability, and (2) how often vulnerabilities are independently discovered, and what the potential implications are of this rediscovery rate.

4.4.1 Time to patch

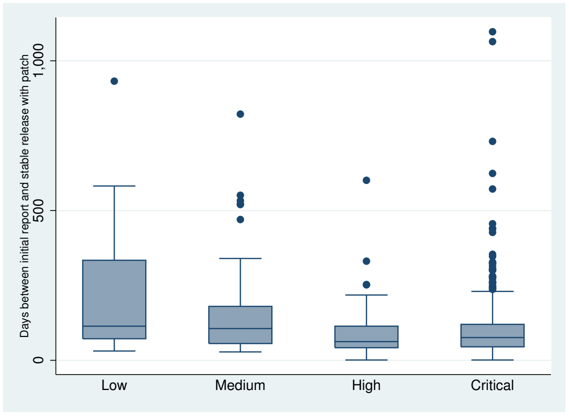
As previously discussed, we choose to study time to release a patched version, *not* time to commit a patch. Although relying on time to release a patch means we analyze only a subset of the data (Section 3), we believe the time to release a patched version of the browser is the more useful metric for end users. Mozilla Firefox and Google Chrome both follow a rapid-release cycle, with a new release every 6 or 7 weeks [11, 25]. In some cases, browser vendors release an out-of-band (or "chemspill") release for vulnerabilities with active exploits in the wild. Such out-of-band releases are one of the most expensive incidents for software companies, with costs running into millions of dollars [30]. Our metric awards the engineering and management commitment required in choosing to release such versions.

Figure 8 depicts the time between initial report of a vulnerability and the stable release that patches it. Table 6 gives summary statistics for these distributions.

Figure 9 is a scatter plot of the same data, which allows us to see changes in time to patch over time. Figure 10 shows the change in standard deviation of time to patch over time. More specifically, for a given release date, the y-value is the standard deviation for all bugs patched in that release or up to five prior releases. These graphs indicate that the standard deviation in time to patch critical vulnerabilities has slowly dropped for Firefox, while Chrome's time to patch critical vulnerabilities has remained relatively constant over time.



(a) Chrome

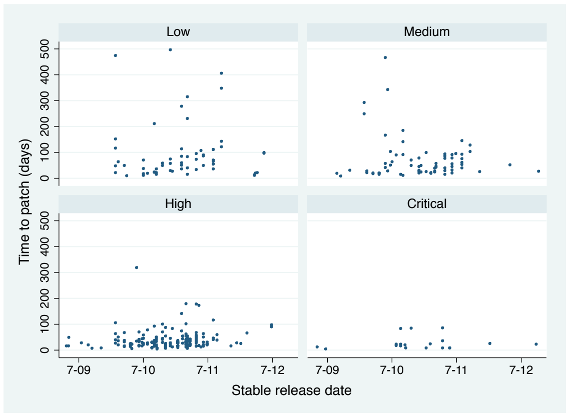


(b) Firefox

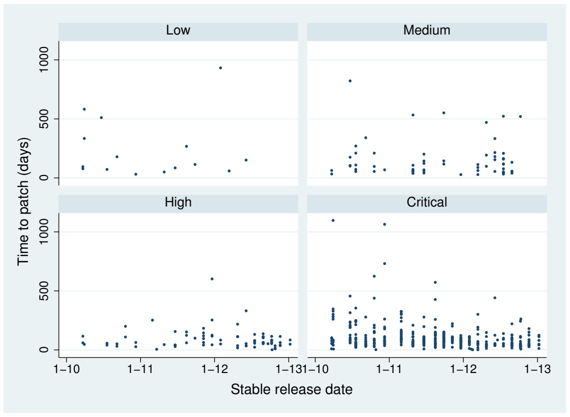
Figure 8: Box and whisker plots depicting the distributions of time between vulnerability report and release that patches the vulnerability, for each severity level.

Discussion For Chrome, both the median time to patch and the variance are lower for higher severity vulnerabilities. This is an important parameter for a VRP because responsible disclosure depends critically on vendor response time [22, 50]. If a vendor does not patch in a reasonable time frame, security researchers are less likely to exercise responsible disclosure. Accordingly, this may be a contributing factor in Firefox's lower degree of community participation (as compared to Chrome), given that the time to patch critical vulnerabilities in Firefox is longer and has very high variance.

In Chrome, the time to patch is faster for critical vulnerabilities than it is for high severity vulnerabilities. This trend continues for medium- and low-severity vulnerabilities as well. This indicates correct prioritization of higher-severity vulnerabilities by Chrome security engineers. The same cannot be said for Firefox; high and critical severity vulnerabilities tend to take about the same amount of time to fix.

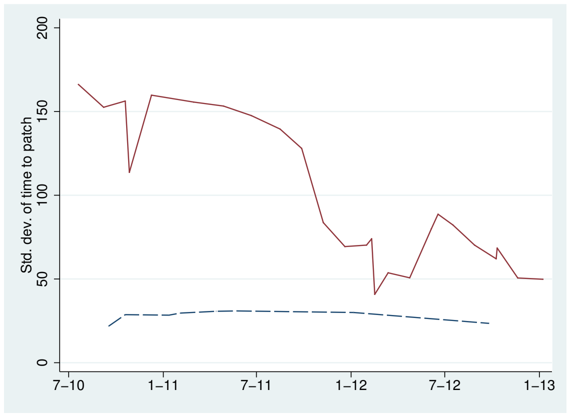


(a) Chrome

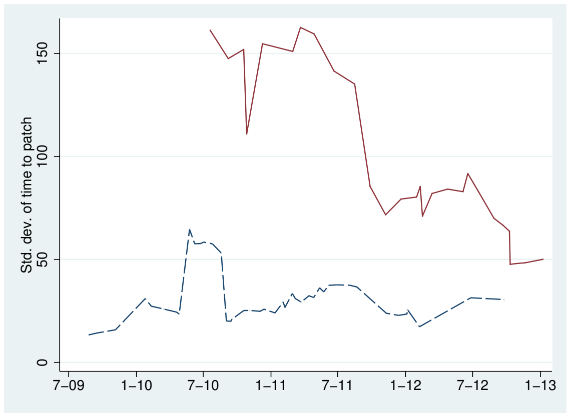


(b) Firefox

Figure 9: Scatter plots depicting the time between vulnerability report and release that patches the vulnerability vs. time, for each severity level.



(a) Critical-severity vulnerabilities.



(b) Critical and high severity vulnerabilities.

Figure 10: Standard deviation of time to patch over time. For a given release date, the y-value is the standard deviation of all bugs patched in that release or up to five prior releases. The red solid line is for Firefox, and the blue dashed line is for Chrome.

The high variance in Firefox's time to patch critical vulnerabilities may be partly attributable to the lack of privilege separation in Firefox, since a larger TCB for critical vulnerabilities means that there is a larger pool of engineers owning code that might hold a critical vulnerability. However, it is an encouraging sign that Firefox has gradually reduced this variance. Nonetheless, the variance in patch times and typical time to patch for Firefox both remain far higher than we see for Chrome, suggesting the need for a concerted effort at reducing this.

4.4.2 Independent discovery

Using the Chromium release blog, we manually coded an additional variable `independent`. This variable represents the number of times a vulnerability was independently discovered. We coded it using the text of the `credit` variable, which mentions "independent discovery" of a vulnerability in the case of multiple independent discoveries.

Our Chrome dataset indicates when a vulnerability was independently discovered by multiple parties, identifies the parties, and in some cases, gives an upper bound on the time between discovery and rediscovery. Of the 668 vulnerabilities in our Chrome VRP dataset, fifteen (2.25%) of them had at least two independent discoveries, and two of these had three independent discoveries. This is a lower bound on the number of independent discoveries of these vulnerabilities, since it represents only those known to the vendor.

Figure 11 displays the independent rediscovery rates for individuals. Each dot represents an individual contributor in our dataset. Its x-value gives the number of vulnerabilities discovered by this individual, and its y-value gives the number of these vulnerabilities independently rediscovered by another contributor in our dataset. Of those individuals who reported five or more vulnerabilities, the highest rediscovery rate is 25% and the mean is

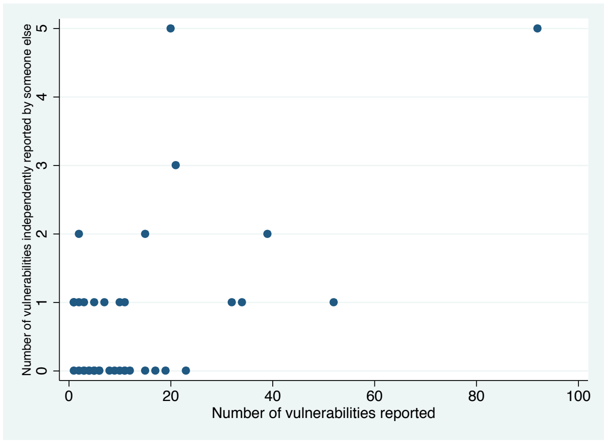


Figure 11: Independent vulnerability discoveries within the Chrome VRP dataset. Each dot represents an individual contributor in our dataset. Its x-value gives the number of vulnerabilities contributed by this individual, and its y-value gives the number of these contributions that were independently discovered by another contributor in our dataset.

4.6%.

Our Firefox dataset does not indicate independent rediscovery, but we have limited data from personal communication with a Firefox security engineer [56]. He indicated that there had been at least 4–7 vulnerabilities reported through the VRP for which there had been two independent discoveries, a rate of 2.7% to 4.7%, which is consistent with what we see in our Chrome dataset.

Discussion Independent rediscovery rates can have implications for estimating the number of latent bugs in software [29] as well as understanding the expected decay rate of a stash of zero-day vulnerabilities.

A zero-day loses its value when the vendor becomes aware of it, which happens via independent discovery of the vulnerability. Thus, a stash of zero-days will decay at some rate. From the limited data available to us via our study, we hypothesize that:

Hypothesis 7 *The decay rate of a stash of zero-day vulnerabilities is low enough to be inconsequential as a result of relatively low empirical independent rediscovery rates.*

We encourage future studies that aim to confirm or refute this hypothesis using larger, more appropriate datasets.

5 Discussion and recommendations

In this section, we synthesize what we have learned and present concrete recommendations for software vendors based on our data analysis.

5.1 Mozilla Firefox vs. Google Chrome

Despite costing approximately the same as the Mozilla program, the Chrome VRP has identified more than three times as many bugs, is more popular and shows similar participation from repeat and first-time participants. There is a stark difference between the levels of external participation in the two VRPs (Figure 2).

Despite having the oldest bounty program, external contributions lag far behind internal contributions to Firefox's security advisories. In contrast, external contributions to Chrome's security advisories closely rival internal contributions. Investigating further, we find three key differences between the two programs:

Tiered structure with large special rewards Mozilla's program has a fixed payout of $3,000, which is approximately equal to the normal maximum payout for Chrome ($3,1337). Nonetheless, Chrome's tiered structure, with even higher payouts (e.g., $10,000) possible for clever bugs and special cases appears to be far more effective in encouraging participation. This makes sense with an understanding of incentives in lotteries: the larger the potential prize amount, the more willing participants are to accept a lower expected return, which, for VRPs, means the program can expect more participants [5].

Time to patch We see a far higher variance in the time-to-release-patch metric for critical vulnerabilities in Mozilla Firefox. It is generally accepted that the viability of responsible disclosure depends on a reasonable vendor response time [50]. Thus, the high variance in Mozilla's response time could affect responsible disclosure through the VRP.

Higher profile Chrome's VRP has a higher profile, with annual competitions like Pwnium providing particularly high rewards (up to $150,000). Chrome authors also provide extra reward top-ups for "interesting" bugs. We believe this sort of "gamification" leads to a higher profile for the Chrome VRP, which may help encourage participation, particularly from researchers interested in wider recognition.

Our methodology does not provide insight into the motivations of security researchers and the impact of VRP designs on the same—a topic we leave for future work. Nevertheless, we hypothesize that these three factors combined explain the disparity in participation between the Firefox and Chrome VRPs. Accordingly, we recommend Mozilla change their reward structure to a tiered system like that of Chrome. We urge Mozilla to do whatever it takes to continue to reduce the variance in time to release a patch for critical vulnerabilities, though we also realize the difficulty involved in doing so. Ongoing attempts at privilege separation might enable reducing the variance in time to patch critical vulnerabilities [17, 36, 39]. Mozilla can also consider holding its own annual competitions or otherwise increasing the PR surrounding its VRP.

5.2 Recommendations for vendors

Our study of the Chrome and Firefox VRPs yield a number of observations that we believe can guide vendors interested in launching or evolving their own VRPs.

We find that VRPs appear to provide an economically efficient mechanism for finding vulnerabilities, with a reasonable cost/benefit trade-off (Sections 4.1.1 and 4.1.6). In particular, they appear to be 2-100 times more cost-effective than hiring expert security researchers to find vulnerabilities. We therefore recommend that more vendors consider using them to their (and their users') advantage. The cost/benefit trade-off may vary for other types of (i.e., non-browser) software vendors; in particular, the less costly a security incident is for a vendor, the less useful we can expect a VRP to be. Additionally, we expect that the higher-profile the software project is (among developers and security researchers), the more effective a VRP will be.

Response time, especially for critical vulnerabilities, is important (Section 4.4.1). High variance in time-to-patch is not appreciated by the security community. It can reasonably be expected to reduce participation because it makes responsible disclosure through the VRP a less attractive option than the other options available to security researchers.

VRP incentive design is important and should be carefully considered. Chrome's tiered incentive structure appears more effective at encouraging community participation than Firefox's fixed-amount incentive structure (Section 4.2.1). Additionally, both Chrome and Firefox have increased their rewards over time. Doing so increases publicity, entices participants, and signals that a vendor is betting that their product has become more secure over time.

Our analysis demonstrates the impact of privilege separation on the Chrome VRP (Section 4.1.2). Privilege separation also provides flexibility to the Chrome team. For example, a simple way for Chrome to cut costs while still increasing participation could be to reduce reward amounts for high-severity vulnerabilities and increase reward amounts for critical-severity vulnerabilities. Mozilla does not have this flexibility. Vendors should consider using their security architecture to their advantage.

6 Related Work

Mein and Evans share our motivation and present Google's experience with its vulnerability rewards programs [35]. In contrast, our focus is on understanding and comparing two popular VRPs run by competing browser vendors. We also perform a number of analyses not performed by the previous work as well as make our data available for other researchers. We also independently confirm that, for both Google and Mozilla, VRPs are cost-effective mechanisms for finding security vulnerabilities.

Development lifecycle datasets Many authors have looked to large datasets, including code repositories, bug trackers, and vulnerability databases, to gather and analyze data in an effort to better understand some aspect of the development lifecycle. Rescorla gathered data from NIST's ICAT database (which has since been updated and renamed to NVD [44]) to analyze whether vulnerability rates tend to decrease over time [49]. He found no evidence that it is in fact worthwhile for software vendors to attempt to find vulnerabilities in their own software because there is no evidence that such efforts are reducing vulnerability rates.

Ozment and Schechter used the OpenBSD CVS repository to ask and answer similar questions as Rescorla [47]. They find that the rate of discovery of what they call *foundational* vulnerabilities—those present since the beginning of the study period—had decreased over the study period.

Neuhaus and Plattner use vulnerability reports for Mozilla, Apache httpd, and Apache Tomcat to evaluate whether vulnerability fix rates have changed over time [42]. They conclude that the supply of vulnerabilities is not declining, and therefore that attackers and/or vulnerability researchers have not hit diminishing returns in looking for vulnerabilities.

Neuhaus et al. use a dataset of Firefox security advisories in combination with the Firefox codebase to map vulnerabilities to software components and predict which components are likely to contain vulnerabilities [43].

Scholte et al. use the NVD to evaluate how cross-site scripting and SQL injection vulnerabilities have evolved over time [52]. They find that the complexity of such vulnerabilities does not appear to have changed over time and that many foundational cross-site scripting vulnerabilities are still being discovered.

Evaluating vulnerability-finding techniques Other work has focused specifically on evaluating the many available techniques for finding vulnerabilities, though we are unaware of any previous work that has considered public-facing VRPs as one such technique.

Austin and Williams evaluated four different techniques for vulnerability discovery on two health record systems: "systematic and exploratory manual penetration testing, static analysis, and automated penetration testing" [2], finding that very few vulnerabilities are in fact found by multiple techniques and that automated penetration testing is the most effective in terms of vulnerabilities found per hour.

Finifter and Wagner compared manual source code analysis to automated penetration testing on a web application, with similar findings: the techniques are complementary, and manual analysis found more vulnerabilities, but took much more time than automated penetration testing [24].

Edmundson et al. found that different reviewers tend to find different vulnerabilities and, even in a small codebase, it takes many reviewers to spot all or even a significant fraction of the vulnerabilities present [18]. This is consistent with our findings about the effectiveness of crowdsourced VRPs.

A large body of work investigates defect prediction using empirical techniques; we refer the reader to a survey by Catal et al. [10].

7 Conclusion and future work

We examined the characteristics of well-known vulnerability rewards programs (VRPs) by studying two such VRPs. Both programs appear economically efficient, comparing favorably to the cost of hiring full-time security researchers. The Chrome VRP features low expected payouts accompanied by high potential payouts, a strategy that appears to be effective in engaging a broad community of vulnerability researchers.

We hope that our study of these two VRPs serves as a valuable reference for software vendors aiming to evolve an existing VRP or start a new one. Potential future work on understanding VRPs includes economic modeling of VRPs; identifying typical patterns, trajectories, or phases in a VRP; and studying failed or unsuccessful VRPs to get a better sense of possible pitfalls in VRP development. Gathering and analyzing data from more VRPs will surely paint a more complete picture of their potential costs and benefits.

Acknowledgments

We are particularly grateful to Chris Evans and Dan Veditz for their help, encouragement, and feedback throughout the research. We also thank Chris Hofmann, Parisa Tabriz, Vern Paxson, Adrienne Felt, the anonymous reviewers, and our shepherd, Sam King, for their feedback on drafts of the paper.

This work was supported by Intel through the ISTC for Secure Computing; by the National Science Foundation under a Graduate Research Fellowship and grant numbers CCF-0424422, 0842695, and 0831501-CT-L; by the Air Force Office of Scientific Research under MURI awards FA9550-08-1-0352, FA9550-09-1-0539, and FA9550-12-1-0040; and by the Office of Naval Research under MURI grant no. N000140911081. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF, the AFOSR, the ONR, or Intel.

References

[1] ADAMSKI, L. Refresh of the Mozilla Security Bug Bounty Program, July 2010. `https://blog.mozilla.org/security/2010/07/15/refresh/`.

[2] AUSTIN, A., AND WILLIAMS, L. One technique is not enough: A comparison of vulnerability discovery techniques. In *Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on* (2011), IEEE, pp. 97–106.

[3] BARRETT, M. PayPal "Bug Bounty" Program for Security Researchers, June 2012. `https://www.thepaypalblog.com/2012/06/paypal-bug-bounty-program/`.

[4] BARTH, A., JACKSON, C., REIS, C., AND TEAM, T. G. C. The Security Architecture of the Chromium Browser. Tech. rep., Stanford University, 2008.

[5] BHATTACHARYYA, N., AND GARRETT, T. A. Why People Choose Negative Expected Return Assets - An Empirical Examination of a Utility Theoretic Explanation. *Federal Reserve Bank of St. Louis Working Paper Series* (March 2006). `http://research.stlouisfed.org/wp/2006/2006-014.pdf`.

[6] BLINDU, E. Vulnerabilities reward programs, July 2012. `http://www.testalways.com/2012/07/13/vulnerabilities-reward-programs/`.

[7] BUCHANAN, K., EVANS, C., REIS, C., AND SEPEZ, T. A Tale of Two Pwnies (Part 2), June 2012. `http://blog.chromium.org/2012/06/tale-of-two-pwnies-part-2.html`.

[8] BUCHANAN, K., EVANS, C., REIS, C., AND SEPEZ, T. Show off Your Security Skills: Pwn2Own and Pwnium 3, January 2013. `http://blog.chromium.org/2013/01/show-off-your-security-skills-pwn2own.html`.

[9] CARETTONI, L. "No More Free Bugs" Initiative, October 2011. `http://blog.nibblesec.org/2011/10/no-more-free-bugs-initiatives.html`.

[10] CATAL, C., AND DIRI, B. A systematic review of software fault prediction studies. *Expert Systems with Applications 36*, 4 (2009), 7346–7354.

[11] Chromium Development Calendar and Release Info. `http://www.chromium.org/developers/calendar`.

[12] Severity Guidelines for Security Issues. `https://sites.google.com/a/chromium.org/dev/developers/severity-guidelines`.

[13] Chromium Bug Tracker, 2013. `http://crbug.com`.

[14] Security: Pwnium 2 tcmalloc profile bug, 2012. `http://crbug.com/154983`.

[15] DAVIS, N. Secure Software Development Life Cycle Processes, July 2012. `https://buildsecurityin.us-cert.gov/bsi/articles/knowledge/sdlc/326-BSI.html`.

[16] Defense in Depth. `http://www.nsa.gov/ia/_files/support/defenseindepth.pdf`.

[17] MozillaWiki: Electrolysis, April 2011. `https://wiki.mozilla.org/Electrolysis`.

[18] EDMUNDSON, A., HOLTKAMP, B., RIVERA, E., FINIFTER, M., METTLER, A., AND WAGNER, D. An Empirical Study on the Effectiveness of Security Code Review. In *Proceedings of the International Symposium on Engineering Secure Software and Systems* (March 2013).

[19] EVANS, C. Celebrating Six Months of Chromium Security Rewards, July 2010. `http://blog.chromium.org/2010/07/celebrating-six-months-of-chromium.html`.

[20] EVANS, C. Bug bounties vs. black (& grey) markets, May 2011. `http://scarybeastsecurity.blogspot.com/2011/05/bug-bounties-vs-black-grey-markets.html`.

[21] Evans, C. Personal Communication, March 2013.

[22] Evans, C., Grosse, E., Mehta, N., Moore, M., Ormandy, T., Tinnes, J., Zalewski, M., and Team, G. S. Rebooting Responsible Disclosure: a focus on protecting end users, July 2010. http://googleonlinesecurity.blogspot.com/2010/07/rebooting-responsible-disclosure-focus.html.

[23] Evans, C., and Schuh, J. Pwnium: rewards for exploits, February 2012. http://blog.chromium.org/2012/02/pwnium-rewards-for-exploits.html.

[24] Finifter, M., and Wagner, D. Exploring the relationship between web application development tools and security. In *USENIX conference on Web application development* (2011).

[25] MozillaWiki: RapidRelease/Calendar, January 2013. https://wiki.mozilla.org/RapidRelease/Calendar.

[26] Fisher, D. Microsoft Says No to Paying Bug Bounties, July 2010. http://threatpost.com/microsoft-says-no-paying-bug-bounties-072210/.

[27] Chrome Releases: Stable Updates. http://googlechromereleases.blogspot.com/search/label/Stable%20updates.

[28] Gorenc, B. Pwn2Own 2013, January 2013. http://dvlabs.tippingpoint.com/blog/2013/01/17/pwn2own-2013.

[29] Hatton, L. Predicting the Total Number of Faults Using Parallel Code Inspections. http://www.leshatton.org/2005/05/total-number-of-faults-using-parallel-code-inspections/, May 2005.

[30] Hofmann, C. Personal Communication, March 2013.

[31] Holler, C. Trying new code analysis techniques, January 2012. https://blog.mozilla.org/decoder/2012/01/27/trying-new-code-analysis-techniques/.

[32] Creating a Computer Security Incident Response Team: A Process for Getting Started, February 2006. https://www.cert.org/csirts/Creating-A-CSIRT.html.

[33] Matthew Finifter and Devdatta Akhawe and David Wagner. Chrome and Firefox VRP datasets, June 2013. https://gist.github.com/devd/a62f2afae9f1c93397f5.

[34] The MEGA Vulnerability Reward Program, February 2013. https://mega.co.nz/#blog_6.

[35] Mein, A., and Evans, C. Dosh4Vulns: Google's Vulnerability Reward Programs", March 2011.

[36] Melven, I. MozillaWiki: Features/Security/Low rights Firefox, August 2012. https://wiki.mozilla.org/Features/Security/Low_rights_Firefox.

[37] Miller, C. The legitimate vulnerability market: the secretive world of 0-day exploit sales. In *WEIS* (2007).

[38] Mills, E. Facebook launches bug bounty program, July 2011. http://news.cnet.com/8301-27080_3-20085163-245/facebook-launches-bug-bounty-program/.

[39] Mozilla Bugzilla. Bug 790923: Content process sandboxing via seccomp filter. https://bugzil.la/790923.

[40] Mozilla Foundation. Mozilla Foundation Security Advisories, January 2013. https://www.mozilla.org/security/announce/.

[41] Netscape announces "netscape bugs bounty" with release of netscape navigator 2.0 beta. The Internet Archive. http://web.archive.org/web/19970501041756/www101.netscape.com/newsref/pr/newsrelease48.html.

[42] Neuhaus, S., and Plattner, B. Software security economics: Theory, in practice. In *WEIS* (2012).

[43] Neuhaus, S., Zimmermann, T., Holler, C., and Zeller, A. Predicting vulnerable software components. In *Proceedings of the 14th ACM conference on Computer and communications security* (2007), ACM, pp. 529–540.

[44] National Vulnerability Database. http://nvd.nist.gov/.

[45] Obes, J. L., and Schuh, J. A Tale of Two Pwnies (Part 1), May 2012. http://blog.chromium.org/2012/05/tale-of-two-pwnies-part-1.html.

[46] Understanding Operational Security. http://www.cisco.com/web/about/security/intelligence/opsecurity.html.

[47] Ozment, A., and Schechter, S. E. Milk or wine: does software security improve with age. In *In USENIX-SS06: Proceedings of the 15th conference on USENIX Security Symposium* (2006), USENIX Association.

[48] Raymond, E. S. *The Cathedral and the Bazaar*, 1st ed. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 1999.

[49] Rescorla, E. Is finding security holes a good idea? *IEEE Security & Privacy 3*, 1 (2005), 14–19.

[50] CERT/CC Vulnerability Disclosure Policy, November 2012. https://www.cert.org/kb/vul_disclosure.html.

[51] Chromium bug tracker: Sandbox bypasses found in review, 2013. http://goo.gl/13Z1R.

[52] Scholte, T., Balzarotti, D., and Kirda, E. Quo vadis? a study of the evolution of input validation vulnerabilities in web applications. *Financial Cryptography and Data Security* (2012), 284–298.

[53] Secunia Vulnerability Coordination Reward Program (SVCRP). https://secunia.com/community/research/svcrp/.

[54] The BlueHat team. Microsoft Security Bounty Programs. http://www.microsoft.com/security/msrc/report/bountyprograms.aspx, June 2013.

[55] The Chromium Authors. Vulnerability Rewards Program: Rewards FAQ, 2010. http://goo.gl/m1MdV.

[56] Veditz, D. Personal Communication, February 2013.

[57] Vulnerability Remediation, September 2010. https://www.cert.org/vuls/remediation.html.