Given Enough Eyeballs, All Bugs Shallow?

Revisiting Eric Raymond with Bug Bounty Programs

Thomas Maillart[1], Mingyi Zhao[2], Jens Grossklags[2], and John Chuang[1]

[1] University of California, Berkeley
School of Information
102 South Hall
Berkeley, CA 94720
[2] PennState University
College of Information Science and Technology
329A Information Sciences and Technology Building
University Park, PA 16802

Abstract. Bug bounty programs offer a modern platform for organizations to crowdsource their software security and for security researchers to be fairly rewarded for the vulnerabilities they find. Little is known however on the incentives set by bug bounty programs: How they drive new bug submissions, and how they improve security through the progressive exhaustion of discoverable vulnerabilities. Here, we recognize that bug bounty programs create tensions for organizations running them on the one hand, and for security researchers on the other hand. At the level of one bug bounty program, security researchers face a sort of St-Petersburg paradox: The probability of finding additional bugs decays fast, and thus can hardly be matched with a sufficient increase of monetary rewards. Furthermore, bug bounty program managers have an incentive to gather the largest possible crowd to ensure a larger pool of expertise, which in turn increases competition among security researchers. As a result, we find that researchers have high incentives to switch to newly launched programs, for which a reserve of *low-hanging fruit* vulnerabilities is still available. Our results inform on the technical and economic mechanisms underlying the dynamics of contributions, and may in turn help improve the mechanism design of bug bounty programs that get increasingly adopted by cybersecurity savvy organizations.

1 Introduction

On March 2nd, 2016, the Pentagon announced the launch of its first *bug bounty* program [**?**]. From now on, the most paranoid organization in the United States will incentivize hackers to break into its systems and report found vulnerabilities for a reward. Although bug bounty programs have mushroomed in the last few years, this audacious announcement by a prominent defense administration may set a precedent, if not a standard, for the future of cybersecurity practice.

Software security has long been recognized as a *hard* computational problem [**?**], which requires additional human intelligence. However, given today's computer systems complexity, individual human intelligence seems to have become insufficient, and

organizations interested in drastically increasing their security are tempted to tap the wisdom of crowds [**?**], just like other disciplines have found ways to mobilize people at scale for their hard problems, such as for sorting galaxies in astronomy [**?**], folding proteins in biology [**?**], recognizing words from low quality book scans [**?**] or to address outstanding mathematics problems [**?**, **?**].

All the above examples involve various aspects of human intelligence, ranging from pattern recognition (Captcha) to highest abstraction levels (mathematical conjectures). It is not clear what kind of intelligence is required to find bugs and vulnerabilities in software, but it generally requires a high level of programming proficiency coupled with *hacking* skills to piece things apart, in order to understand them better. In a nutshell, searching for complicated bugs and vulnerabilities may be a hard and time-consuming task, which is generally not, or at least no longer, considered as a leisure that hackers perform for hedonic pleasure or for the good.

Therefore, nowadays some (monetary) incentives must be set, in order to get security researchers hunt bugs. Offering rewards for vulnerabilities has been a long endeavor over the last decade [**?**], with many more or less successful attempts to set incentives right [**?**, **?**, **?**]. HackerOne, a leading online service dedicated to help organizations set up and manage their own bug bounty program, has paved the way to the deployment of bounty programs at scale. Nevertheless, in this pioneering era of bug bounty hunting, it remains unclear how current mechanism designs and incentives structures influence the long-term success of bounty programs, and how they may evolve in the foreseable future, following enhanced understanding of bug discovery mechanisms on the one hand [**?**], and on the other hand, from better characterization of the utility functions of respectively (i) organizations operating a bug bounty program and (ii) security researchers.

Here, we have investigated a public data set of 35 public bug bounty programs from the HackerOne website. We find that as more vulnerabilities get discovered within a bounty program, security researchers face an increasingly difficult environment in which the probability of find a bug decreases fast, while reward increases. This phenomenon is reminiscent of the St-Petersburg paradox famous in behavioral economics, in which people have increasingly hard time to make a rational choice. For this reason, as well as because the probability to find a bug decreases faster compared to the payoff increase, security researchers are incentivized to consistently switch to newly launched research programs, at the expense of older programs. This switching phenomenon has already been found in [**?**]. Here, we characterize it further, by quantifying the evolution of incentives as more vulnerabilities get discovered in bug bounty program, and how researchers benefit on the long term from switching to newly launched programs.

This article is organized as follows. Related research is presented in Section **??**. Important features of the data set used here is detailed in Section **??**. We then introduce the main mechanism driving vulnerability discovery in Section **??**. Results are

presented and discussed in respectively Sections **??** and **??**. We finally in conclude in Section **??**.

2 Related work

Achieving software reliability has concerned engineers for at least four decades [**?**, **?**, **?**]. Early empirical work on software bug discovery dates back to the time of UNIX systems [**?**], and over years, numbers of models for discovering vulnerabilities have been developed (see [**?**, **?**] for some of the most contemporary approaches). However, as early as in 1989, it was recognized the time to achieve a given level of software reliability is inversely proportional to the desired frequency level [**?**]. For example, in order to achieve a 10^{-9} probability of failure, a software routine should be tested 10^9 times. Actually, the random variable $P(T > t) = 1/t$ corresponds to the Zipf's law [**?**, **?**], which diverges as the random variable sample increases (i.e., no statistical moment is defined), and thus, it was rightly concluded that there would be software vulnerabilities as long as long enough resources and time could be provided to find them. This problem can also be seen from an entropy maximization perspective, which is good for evolution (e.g., in biology) but detrimental in software engineering [**?**].

To date, no systematic algorithmic approach has been found to get rid of bugs at a speed that would allow following the general pace of software evolution and expansion. Thus human intelligence is still considered as one of the most efficient ways to find bugs, and most importantly to uncover software vulnerabilities. Management techniques and governance approaches have been developed to help software developers and security researchers in their review tasks, starting with pair programming [**?**]. To protect against cyber-criminals, it is also fashionable to hire *ethical hackers*, who have a mindset similar to potential attackers, in order to probe the security of computer systems [**?**, **?**, **?**]. Inherited from the open source philosophy, the full disclosure policy has been hotly debated as promoting a safer Internet [**?**], although in practice it seems that full disclosure has increasingly become a dominant standard, that helps people and organizations get informed as fast as possible about software vulnerabilities. In most of these successful human-driven approaches, there is a knowledge-sharing component, may it be between two programmers sitting together in front of a screen, ethical hackers being hired to discover and explore the weaknesses of a computer system, or the broader community being exposed to open source code and publicly disclosed software vulnerabilities. Thus, Eric Raymond's famous quote "Given enough eyeballs, all bugs shallow" [**?**], tends to hold, even though in practice things are often slightly more complicated [**?**].

Recognizing the need of human intelligence for tackling security bugs at scale, researchers have considered early on the importance of trading bugs and vulnerability as a valuable – and often hardly earned – knowledge. *Vulnerability markets* have thus emerged as a way to ensure appropriate incentives for knowledge transfer from security researchers to software and Internet organizations [**?**], and in particular, to jointly harness the wisdom of crowds and reveal the security level of organizations through a

competitive incentive scheme [**?**]. The efficiency of vulnerability markets has however been nevertheless questioned on both theoretical [**?**, **?**] and empirical grounds [**?**, **?**].

Early on and building on previous work by Schechter [**?**], Andy Ozment [**?**] recognized that in theory most efficient mechanism designs shall not be markets *per se*, but rather auction systems [**?**]. In a nutshell, the proposed (monopsonistic) auction mechanism implies an initial $R(t = t_0) = R_0$, which increases linearly with time. If a vulnerability is reported more than once, only the first reporter receives the reward. Therefore, security researchers have an incentive to submit a vulnerability early (before other researchers might submit the same vulnerability), but not too early, so that can maximize their payoff $R(t) = R_0 + \epsilon \cdot t$ with ϵ the linear growth factor, which is also supposed to compensate for the increasing difficulty of finding each new bug. But setting the right incentive structure $\{R_0, \epsilon\}$ is not trivial, because it must account for number of uncertainties [**?**], such as work needed, or effective competition (i.e., the number of researchers enrolled in the bug program).

Regardless of theoretical considerations (or perhaps by integrating them), vulnerability reward programs have emerged as a tool used by the industry, first launched by specific software companies for their own needs and with rather heterogeneous incentive schemes [**?**], including with no monetary reward [**?**], and followed by dedicated platforms comparable to trusted third parties in charge of clearing transactions between bug bounty programs launched by organizations and security researchers. These platforms also assist organizations in the design and deployment of their own program. The currently leading platform is HackerOne.[3] HackerOne runs 35 public programs, for organizations across a wide range of business sectors, and for which bounty awards are reported on their website (and a non-disclosed amount of private programs). Previous research has investigated vulnerability trends, response & resolve behaviors, as well as reward structures of participating organizations. In particular, it was found that a considerable number of organizations exhibit decreasing trends for reported vulnerabilities, yet monetary incentives have a significantly positive correlation with the number of vulnerabilities reported [**?**].

3 Data

The data were collected from the public part of the Hacker One website. From 35 public bounty programs, we collected the rewards received by security researchers (in US dollars), with their timestamps. Since HackerOne started its platform in December 2013, new programs have been launched roughly every two months, following an essentially memoryless Poisson process ($\lambda = 57$ days, $p < 0.001$ and $R^2 > 0.99$). Figure **??**A shows the timeline of the 9 most active programs with at least 90 (rewarded) bug discoveries, as of February 15, 2016. When a new program is launched, we observe an initial peak (or within weeks after launch), which accounts for the majority of discoveries, suggesting a *windfall* effect. Following the initial surge of vulnerability discoveries,

[3] HackerOne, `https://hackerone.com/` (last access March, 4th 2016).

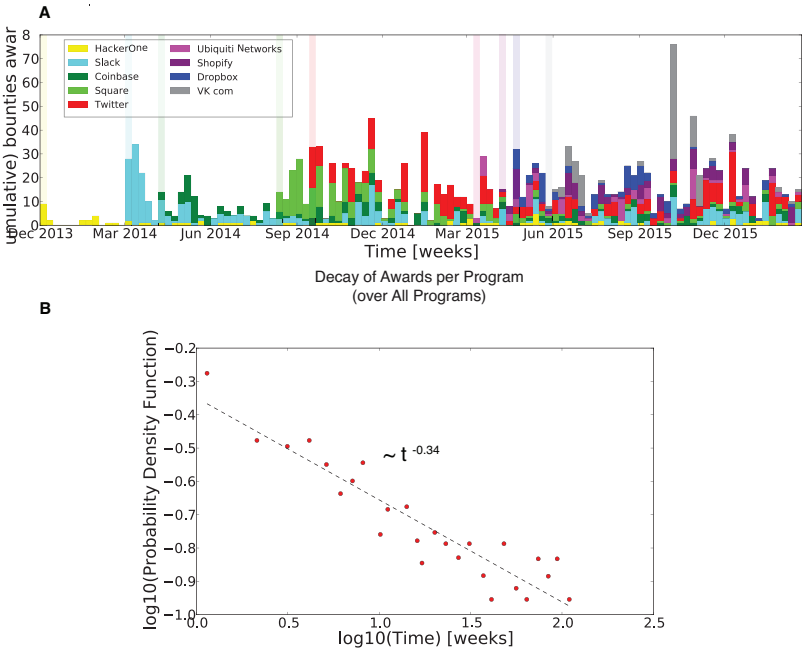

Fig. 1. A. Weekly vulnerability discoveries for the 9 most active programs (with at least 90 bug discoveries as of February 15, 2016). The light colored vertical bars represent the start of the program, occurring when the first bounty is awarded. Most programs exhibit an initial shock (or shortly after the start of the program), followed by a decay of discoveries, which is characterized at the aggregate level by a long-memory process (panel **B**) characterized by a power law decay $\sim t^{\alpha}$ with $\alpha = -0.40(4)$ ($p < 0.001$ and $R^2 = 0.79$). Each data point in the figure is the median of normalized vulnerability numbers of all 35 programs considered in this study.

bounty awards become less frequent following a decay function with long-memory, following a robust power law decay $\sim t^{\alpha}$ with $\alpha = -0.40(4)$ ($p < 0.001$ and $R^2 = 0.79$) at the aggregate level and over all 35 bounty programs, with weekly binned time series of bug discoveries normalized by their initial value (see Figure **??**B).

The long-memory process of bug discovery following the launch of a bounty program we observe here, is reminiscent of human timing effects: When the program launches, it takes some time first for the researcher to be exposed to the new program (through the media and social media); Second for the researcher to find and submit bugs, and third for the organization managing the bug bounty program to assess the quality of each submission, and assign a proper reward. To account for all these delays, one may resort to priority queueing applied to humans: First, competing attention prevents immediate exposure to the news of a new program; Second, when security researchers get interested in a new program, they may still be actively searching bugs on other programs or performing other tasks, such as e.g., their regular job, leisure, family

matters); Third, when subjected to a flow of bug submissions, security teams at organizations leading bounty programs assign priorities among submissions, and resolve them with human resources available at the time of submission. These delays are best rationalized by human timing contingencies, and moreover, by time as a scare, non-storable resource, which are known to generate long-memory responses of the form $\sim t^{-1.5}$ (in fully rational case) between the arrival and the execution of a task [**?**]. The observed much slower decay may result from the compound effect of multiple delays, such as those mentioned above. The initial burst of discoveries, followed by a long-memory decay may also result from the increasing difficulty associated with finding new bugs for each bounty program, as the most obvious vulnerabilities get uncovered first. Since, we consider only the *time of discovery* as the moment when the validity of the bug submitted is acknowledged by the program manager, we are mostly blind to the human timing effects associated with the long-memory process observed on Figure **??**B, including when submissions are made, but don't lead to a discovery associated with a monetary reward.

4 Method

Bug bounty programs work on the premise that humans are efficient at searching and finding vulnerabilities, in particular when large pools of security researchers with a variety of skills can be mobilized for the task. It is in the interest of the organization launching a bounty program to *exhaust* vulnerabilities, or to reduce the probability of finding additional vulnerabilities to a residual level. In addition, incentives must be carefully set. Here, we investigate the interplay between the vulnerability exhaustion process, and the cumulative reward distributed to security researchers within and across bounty programs. When a bug bounty program starts, it attracts a number of security researchers, who in turn submit bugs. Subsequent bug discoveries get increasingly difficult [**?**], and program managers must reward vulnerabilities accordingly in order to keep security researchers onboard (or to attract new ones according to the current level of difficulty).

Starting from an initial probability of discovering the first vulnerability $P(k=0)=1$, we assume that the probability to find a second (and subsequent) vulnerability(ies), is a fraction of the former probability: $P(k+1)=\beta * P(k)$ with β a constant strictly smaller than, yet usually close to 1. The probability that no more discovery will be made after k steps is given by $P(k)=\beta^k(1-\beta)$. Conversely, starting from the initial reward $R_0=R(k=0)$, the subsequent reward $R_1=\Lambda_1 \cdot R_0$, and further additional reward $R_2=\Lambda_2\Lambda_1 \cdot R_0$. After n steps, the total reward is the sum of all past rewards:

$$R_n = R_0 \sum_{k=1}^{n} \Lambda_1 ... \Lambda_k. \tag{1}$$

Thus, R_n is the recurrence solution of the Kesten map ($R_n=\Lambda_n R_{n-1}+R_0$) [**?**,**?**]: As soon as amplification occurs (technically, some of the factors Λ_k are larger than 1),

the distribution of rewards is a power law, whose exponent μ is a function of β and of the distribution of the factors Λ_k. In the case where all factors are equal to Λ, this model predicts three possible regimes for the distribution of rewards (for a given program): thinner than exponential for $\Lambda < 1$, exponential for $\Lambda = 1$, and power law for $\Lambda > 1$ with exponent $\mu = |\ln \beta| / \ln \Lambda$ (see Appendix). The expected payoff of vulnerability discovery is thus given by,

$$U_k = P_k \times R_k, \tag{2}$$

with both P_k and R_k random variables respectively determined by β and Λ. The nature of U is reminiscent of the St. Petersburg paradox (or St. Petersburg lottery), proposed first by the Swiss Mathematician Nicolas Bernoulli in 1713, and later formalized by his brother Daniel in 1738 [**?**]. The St. Petersburg paradox states the problem of decision-making when both the probability and the reward are diverging, yet the expected utility increases (in the simplest case proposed by Bernoulli, $U_n = \sum_{k=0}^{n} U_k = n$) [**?**]. $U(k)$ therefore determines the incentive structure for security researchers, given that the bounty program manager can tune R_0 and in some cases, the manager can also tune Λ. However, in general rules are set upfront and shall not be changed in the course of the bounty program. Changing game rules is risky as it may undermine trust in the program. Here, we assume the bounty program managers don't tune their reward policy after the bug bounty program has started. In principle, the manager could set R_0 to influence P_0 and indirectly P_k. Mapping the discovery rank k into the rate of discovery may also allow consider discounting aspects in presence of competing opportunities and inter-temporal choices under uncertainty [**?**]. A new bounty program is launched at a Poisson rate, approximately every 2 months, and each launch brings its windfall effect, leaving the researcher with the choice to either keeping digging increasingly harder vulnerabilities (rare but with higher reward), or turning to the low hanging fruits (frequent but with low reward) of a new program. We shall therefore verify whether newly launched programs actually influence security researchers.

5 Results

The vulnerability discovery process in a bug bounty program is driven by the probability to find an additional bug given that k bugs have already been discovered (i.e., the exhaustion process), and program managers aim to maximize n, the total number of bugs found. Our results show that the number of bugs discovered is a super-linear function of security researchers who have enrolled in the program (see Figure **??**A). While bounty programs benefit from the work of a large number of researchers, researchers overall benefit from diversifying their efforts across programs (see Figure **??**C). This benefit is particularly tangible regarding the cumulative reward they can extract from their bug hunting activity. In particular, we illustrate how researchers take the strategic decision to enroll in a newly launched program, at the expense of existing ones. We observe that behaviors seem to contradict rational behavior dictated by the expected utility function, characterized by a structure comparable to the St. Petersburg.

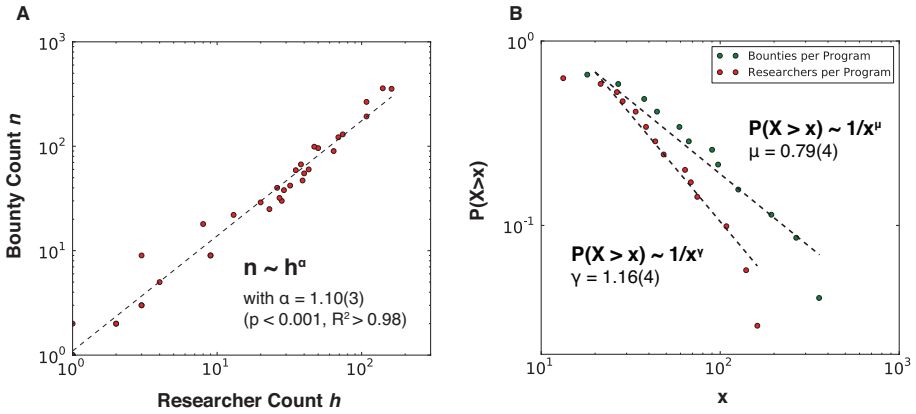

Fig. 2. A. The number of bounty discoveries B_c scales as h^α with $\alpha = 1.10(3)$ and h the number of security researchers enrolled in a program. Since $\alpha > 1$, a bounty programs benefits in a super-linear fashion to the enrollment of more researchers. **B.** The tail distribution of bounty discoveries follows a power law distribution $P(X > x) \sim 1/x^\mu$ with $\mu = 0.79(4) < 1$, with no statistical moment defined (in particular no mean and no variance defined). Thus, from **A.** and **B.** combined and unless a hard limit exists, the number of vulnerabilities which can be found is only limited by the number of researchers enrolled in the program. Furthermore, each additional researcher brings a marginally increasing chance to find a lot more vulnerabilities.

5.1 Security researcher enrollment is determinant for the success of a bug bounty program

As presented on Figure **??**A, we find that the number n of vulnerabilities discovered scales as $n \sim h^\alpha$ with $\alpha = 1.10(3)$ and h the number of security researchers enrolled in a program. Since $\alpha > 1$, a bounty program benefits in a super-linear fashion from the enrollment of more researchers. This result is reminiscent of productive bursts and critical cascades of contributions in open source software development [**?**]: Each enrollment (i.e., *mother* event) initiates a cascade of vulnerability discoveries (i.e., *daughter* events) by the security researcher. Over all participants, the more productive these cascades, the better. The total amount of bug discoveries within a program is a random variable governed by a power law tail distribution of security researchers enrolled with exponent $1 < \gamma = 1.16(4) < 2$ (see Figure **??**B), and by the *fertility* of researchers' investigations (not shown here). In case of critical fertility, i.e., if on average 1 discovery leads to 1 follow-up discovery, the distribution of the sum of bug discoveries follows a power law tail distribution with exponent $\mu = 1/\gamma$ [**?**]. Here, we find that $\mu = 0.79(4) \approx 1/1.16 = 0.86(4)$ in reasonably good accordance with the theory (see Figure **??**B).

Since the tail distribution of bounty discoveries follows a power law distribution $P(X > x) \sim 1/x^\mu$ with $\mu = 0.79(4) < 1$, no statistical moment is defined (in particular neither the mean nor the variance are defined). Therefore, we can expect that in

the future some bounty programs will lead to the discovery of arbitrary large numbers n of bugs. And this large n is controlled by the number of enrolled researchers: The intuition is simply that the more researchers are enrolled, the more likely a particularly productive researcher will be sampled. This highly skewed productivity may result from previously acquired skills, fast endogenous learning, pure chance, or perhaps a combination of multiple factors.

5.2 Security researchers are incentivized to diversify their contributions across bug bounty programs

From the perspective of a security researcher, the governing metric is primarily the number of vulnerabilities found by herself not or over a whole bounty program, but rather the expected payoff from the accumulation of bounty awards over all programs. This expected payoff is governed by the probability to find a given number of vulnerabilities and their associated payoff, as discussed in Section **??**. To fully understand the incentive mechanisms at work, we consider 3 perspectives: (i) the expected cumulative down-payment made by bug bounty program managers (see Figure **??**A), the expected cumulative payoff from the viewpoint of a researcher for (ii) one program (see Figure **??**B), and for (iii) all programs (see Figure **??**C).

The average cumulative down-payment per program exhibits a super-linear scaling as $\sim k^{1.27}$ ($p < 0.001$, $R^2 > 0.99$), while the frequency of vulnerabilities P_k is only slightly upwards trended increasing as $\sim k^{0.13}$ ($p < 0.001$, $R^2 = 0.40$). The expected down-payment by bug bounty program managers therefore scales as $\sim k^{1.40}$. This is a considerable super-linear increase (as $k \to \infty$), which casts questions on the long-term sustainability of bug bounty programs.

From the viewpoint of the researcher and her expected payoff from a single bug bounty program, the increase of average cumulative reward ($R_k \sim k^{1.40}$) does not offset the fast decay of probability ($P_k \sim k^{-1.85}$) to find a vulnerability of rank k. The expected payoff therefore follows $\sim k^{-0.45}$, which clearly does not bring high incentives to explore in depth a bug bounty program. It is important to note however, that the bug bounty manager cannot fix P_k, which is a genuine feature of bug discovery by humans. To maintain positive individual incentives, the manager should set an incremental reward such that $R_k \sim k^{\alpha}$ with $\alpha > 1.40$, which in turn would worsen the down-payment function both in terms of incremental expenditures and in exploration of higher ranks. This approach does not consider possible individual human hard limits, preventing finding additional bugs. In that latter case, setting higher reward incentives would have no effect.

Security researchers tend to switch from one bounty program to another program [**?**, **?**]. The strategy can be assimilated to portfolio diversification [**?**]. Over all bug bounty programs, security researchers have another much more favorable expected payoff: The reward scaling is smaller ($R_k \sim k^{1.24}$ with $p < 0.001$, $R^2 > 0.99$), yet the frequency of bug discoveries decays much slower as a function of rank $P_k \sim k^{-1.05}$ ($p < 0.001$, $R^2 = 0.85$). Therefore, over all bounty programs, security researchers

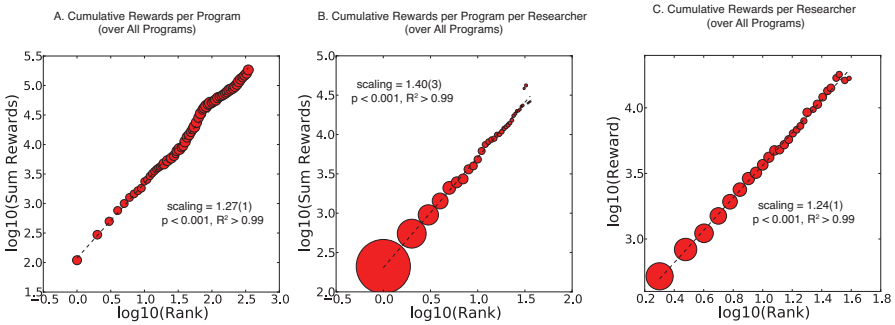

Fig. 3. A. (Log-)binned cumulative down-payment per program over all public programs on the HackerOne platform, scales as $R_k \sim k^{1.27}$ ($p < 0.001$, $R^2 > 0.99$), with k the rank. Each log-bin shows the mean value and the circle sizes depict the number of values in each bin (i.e., the rank frequency). The super-linear *scaling* relationship between the cumulative reward and the rank shows that reward increases as a function of k. However, the frequency of vulnerabilities P_k is only slightly upwards trended increasing as $\sim k^{0.13}$ ($p < 0.001$, $R^2 = 0.40$). **B.** Considering the opposite figure from the viewpoint of researcher's expected payoff when enrolling in a single bug bounty program, the super-linear effect is much stronger ($R_k \sim k^{1.40}$ with $p < 0.001$ and $R^2 > 0.99$). However, the frequency decays following a power law of the form $P_k \sim k^{-1.85}$ ($p < 0.001$, $R^2 = 0.97$). **C.** Over all bug bounty programs, security researchers have another expected payoff: the reward scaling is smaller ($R_k \sim k^{1.24}$ with $p < 0.001$, $R^2 > 0.99$), yet the frequency of bug discoveries decays much slower as a function of rank $P_k \sim k^{-1.05}$ ($p < 0.001$, $R^2 = 0.85$).

have an increasing, yet marginally decreasing, incentive to explore higher ranks as $U_k \sim k^{0.19}$. In a nutshell, security researchers have an incentive to keep searching for bugs on a large variety of bug bounty programs.

5.3 Influence of Newly Launched Programs on Researcher Behaviors

As security researchers weigh their strategic choice to switch their attention from one program to another, the time factor is determinant because the expected payoff is dependent on the current vulnerability rank, which maps into the time dimension (i.e, the duration between two discoveries is drawn from a characteristic random variable, which is not considered here). While switching decision may be made by a researcher at any time, the most obvious moment is when a new program is being launched. Determining the power of attraction of a new program informs on the sudden incentive shift triggered by a punctual event. A number of factors may however influence this incentive shift: For example the notoriety of the organization launching a program in comparison to others may also influence the choices made by researchers. Here, we aim to test 3 hypotheses:

– **H1:** The number of newly launched programs is negatively associated with contributions to existing programs,

– **H2:** Bounty rewards provided by newly launched programs is negatively associated with contributions to existing programs,
– **H3:** The number of newly launched programs have a larger impact on the contribution to older programs.

We use a simple ordinary least square (OLS) regression model, which is specified as follows:

$$V_{it} = \beta_0 + \beta_1 dP_t + \beta_2 T_{it} + \beta_3 A_{it} + \beta_4 B_{it} + \epsilon_{it}. \quad (3)$$

V_{it} is the number of vulnerability reports received by bounty program i in the month t. dP_t is the number of new programs launched in month t. T_{it} is the number of months since bounty program i launched. We first incorporate A_i the log of the Alexa rank, which measures web traffic as a proxy of popularity for organization i. B_i is the log of the average amount of bounty paid by bounty program i. Finally, ϵ_{it} is the unobservable error term. In models **2**-**4**, we extend the basic model to further study competition occurring between bounty programs. These alternative specifications include:

– **Average bounty of newly launched programs:** Intuitively, if new programs offer higher rewards, they should attract more researchers from existing programs. We calculate the average bounty for all new programs in month t as NB_t.
– **Interaction between dP_t and T_{it}:** Conceivably, the effect of new programs on existing programs depends on certain characteristics of the latter, such as age. In particular, we ask if a new entrant has more negative effects on older programs compared to younger programs? To examine this, we consider an interaction term between the number of new programs (dP_t) and the age of the program (T_{it}).
– **Program fixed effect:** To better control for program-specific, time-invariant characteristics, e.g., reputation among researchers, we add program fixed effect in model **4**. The addition of this fixed effect allows us to examine how bug discovery changes over time within each program i.

Consistent with our prediction, the coefficient of dP_t is negative in all 4 specifications. Ceteris paribus, the launch of new programs reduces strongly the number of vulnerabilities reported to existing programs. In other words, the entry of new programs indeed attracts researcher's attention away from existing programs, which is consistent with fast decreasing expected payoff for individuals searching bugs for a specific program. Also, the average bounty paid by new programs ($B_{new,t}$) has a negative effect on existing programs as well, but the coefficient is only significant in model **4**. Again this result is consistent with the theory and above results, as researchers have a high incentive to find the low rank *low-hanging* fruits.

The interaction coefficients for term $T_{it} \times dP_t$ in models **3** and **4** are positive. Therefore, the impact of newly launched programs depends on the age of the existing programs: Compared to younger programs, the negative impact of dP_t is smaller for programs with a longer history, i.e., those with larger T_{it}. At first sight, this results may look at odds with the fact that individual expected payoff from a specific program decreases as a function of rank k, and presumably the older a program the more likely it

VARIABLES	(1) V_{it}	(2) V_{it}	(3) V_{it}	(4) V_{it}
dP_t	-1.235***	-1.350***	-2.310***	-1.236**
	(0.305)	(0.327)	(0.603)	(0.515)
A_i	-23.61***	-23.72***	-23.72***	-7.188**
	(2.140)	(2.156)	(2.152)	(3.473)
B_i	16.64***	16.56***	16.75***	-7.414
	(1.311)	(1.315)	(1.339)	(5.698)
T_{it}	-0.690	-0.658	-3.312***	-3.758***
	(0.426)	(0.427)	(1.239)	(1.128)
$B_{new,t}$		-0.0445	-0.0312	-0.0321*
		(0.0280)	(0.0277)	(0.0184)
$T_{it} \times dP_t$			0.106**	0.0755*
			(0.0431)	(0.0406)
Constant	160.2***	170.4***	190.3***	136.5***
	(16.12)	(18.80)	(23.17)	(26.17)
Observations	1,212	1,212	1,212	1,212
R-squared	0.314	0.316	0.319	0.647
Program FE	No	No	No	Yes

Robust standard errors in parentheses
*** p<0.01, ** p<0.05, * p<0.1

has a high rank. Thus, the switching effect should be stronger. Perhaps our OLS regression model is limited in the sense that it does account for the absolute activity (which decreases very slowly as $t \to \infty$, as shown on Figure **??**B), instead of the variation rate.

6 Discussion

Finding bugs in software code is one of the oldest and toughest problems in computer science. While algorithm based approaches have been developed over the years, human verification has remained a prime approach to debugging and vulnerability hunting. Moreover, the idea of getting "enough eyeballs" to inspect the code has been a cornerstone argument for the open source software movement [**?**] along with the full-disclosure argument, as well as vulnerability markets [**?**]. Bug bounty programs are perhaps the latest successful incarnation of markets for trading bugs and vulnerabilities [**?**], which set incentives to disclose early, combined with an increase of payoff for more rare and difficult bugs.

Here, we have found that the number of bugs and vulnerabilities is super-linearly associated with the number of security researchers. However, the distribution of bugs found per researcher is highly skewed, hence suggesting that while many researchers find only few bugs, a hand of bug hunters do particularly well. This result is reminiscent of productive bursts in open source software development [**?**]. Skewed performance may be attributed to critical cascade mechanisms, but its origins remain unknown. Per-

formance within a bug bounty program most likely combines some *ex-ante* fitness, endogenous learning capabilities, and perhaps some pure luck in the bug search process, which all need further detail investigation and disentanglement.

Most security researchers however participate in multiple bug bounty programs, and when a new program is launched, they face the strategic choice of switching program. Our results show that researchers have a decreasing incentive to explore higher ranks within the same program (Figure **??**B), while they have an increasing, yet marginally decreasing, incentive to explore multiple programs (Figure **??**C). We further confirm that researchers tend to switch when new programs are launched. This is a strong signal that researchers make rational choices in the bug hunting environment: They have high incentives to switch quickly to a new program and harvest as fast as possible many frequent bugs with little reward, rather than less frequent yet more endowed vulnerabilities, even though the reward structure of bug bounty programs seems to incentivize generously the reward of high rank (i.e., less probable) vulnerabilities (Figure **??**A).

This result raises questions on some hard limits associated with incentives associated with bug discovery by humans: The disincentive clearly stems from the difficulty for individual researchers to reach high ranks (i.e., $P_k \to 0$ when $k \to \infty$), not from the reward scheme. However, while it may not be a good deal on average for researchers (i.e., following the expected payoff) to pursue exploration of the same program, some specially skilled or lucky researchers have a chance to strive towards high-rank high-yield bounties. Hence, increasing the enrollment mechanically increases the chance to find those special researchers from a larger pool of researchers.

Using *rankings* provides handy insights on the processes governing the vulnerability discovery process, and to some extent, associated incentives. However, the rank is an arbitrary measure of time, which hardly accounts for the effort spent on researching bugs, as well as for discounting effects. For instance, if the time required to find a vulnerability increases with the rank, then the expected payoff shall be discounted accordingly. Other aspects enter the equation: While most submissions occurs early on after the program launch, this is also the moment when an organization might be less prepared to respond to a large flow of tasks, which in turn may trigger priority queueing and contingent delays [**?**]. While some workaround may be envisioned, publicly available data currently limit some desirable investigations, involving timing and discounting effects.

In this study, we have considered the incentive mechanisms at the aggregate level. Managers however organize enrollment, set incentives and tackle the operational pipeline, involving submission reviews and payroll processing. All these aspects, which are unique to each program, may crowd in, or on the contrary crowd out, security researchers from bug bounty programs. In particular their willingness to participate will be affected, but also the amount of effort they are ready to throw in the search of vulnerabilities. It is the hope of the authors to get increasingly fine-grained insights, to compare bug bounty programs, and thus establish benchmarks of most performing programs, in an environment

driven by large deviation statistics, with outlier contributions, and incentives structures, which resemble the St-Petersburg paradox, a well-known puzzle for decision making in behavioral economics.

7 Conclusion

In this paper, we have investigated how crowds of security researchers hunt software bugs and vulnerabilities. We have found that it is essential for managers to design their program, in order to attract and enroll the largest possible number of security researchers. This results stems for the large deviation of contributions per researchers: Increasing the crowd therefore increases the chance to sample a researcher capable of achieving a large number of bug discoveries. Studying the incentive structure of 35 public bug bounty programs launched at a rate of one per month over 2 years, we have found that security researchers have high incentives to rush to newly launched programs, in order to scoop rewards from numerous easy bugs, and as the program ages (and therefore, the probability of finding a vulnerability decreases), switch to another newer "easier" program. Our results suggest that incentives are not generously set by program managers, but rather that it is quickly getting harder for researchers to find vulnerabilities, once the obvious ones have been discovered.

Appendix

Derivations of the Model for homogenous factors $\Lambda_k = \Lambda$ Here, we provide a detailed study of the possible behaviors of the model, also around the critical point $\Lambda = 1$.

Three regimes must be considered:

1. For $\Lambda < 1$, the distribution is given by

$$P_{\Lambda<1}(S \geq s) = (1-\beta)\left(1 - \frac{s}{s_{\max}}\right)^c \ , \quad s_{\max} := \frac{S_0\Lambda}{1-\Lambda} \ , \ c := \frac{\ln\beta}{\ln\Lambda} > 0 \ . \quad (4)$$

   This distribution can be approximated in its central part, away from the maximum possible reward $s_{\max}$, by a Weibull distribution of the form

$$\Pr(\mathrm{S} \geq s) \sim e^{-(s/d)^c} \ . \quad (5)$$

   For $\Lambda \to 1^-$, we have $s_{\max} \to +\infty$ and, for $s \ll s_{\max}$, expression (**??**) simplifies into a simple exponential function

$$P_{\Lambda\to 1^-}(S \geq s) \sim e^{-|\ln(\beta)|s/S_0} \ . \quad (6)$$

2. For $\Lambda = 1$, the distribution of rewards is a simple exponential function since $S_n = nS_0$ is linear in the rank n and the probability of reaching rank n is the exponential $P(n) = \beta^n(1-\beta)$. Actually, the expression (**??**) becomes asymptotical exact as

$$P_{\Lambda=1}(S \geq s) = (1-\beta)e^{-|\ln(\beta)|s/S_0} \ . \quad (7)$$

3. For $\Lambda > 1$, the distribution of rewards is of the form,

$$P_{\Lambda>1}(S \geq s) = \frac{1}{(1+\frac{s}{s*})^c} \ , \quad s^* := \frac{S_0\Lambda}{\Lambda-1} \ , \quad c := \frac{|\ln\beta|}{\ln\Lambda} \ , \quad (8)$$

   which develops to a power law distribution of reward of the form $\Pr(\text{reward} \geq S) = C/S^\mu$ with $\mu = c$, when $\Lambda \to +\infty$.