Response to reviewers

The authors of submission 76, entitled "Given enough eyeballs, all bugs are shallow: Revisiting Eric Raymond with bug bounty programs" would like to thank the reviewers for their valuable feedback. Here, we provide a response to all points raised, and we shortly describe the changes mades to the manuscript.

Reviewer 1:

1. The authors motivate their work with the following claim: "Thus human intelligence is still considered as one of the most efficient ways to find bugs, and most importantly to uncover software vulnerabilities." I disagree with this claim, as there are by now many mature automated methods for finding bugs automatically. Here are some prominent examples: Coverity (Dawson Engler), FindBugs (Bill Pugh), and SLAM were so successful that they became companies or were deployed at Google or Microsoft. Delta Debugging is also popular (textbook, course, deployed tools, etc). Finally, they might look at Astree. (happy to provide pointers to these systems if the authors are not familiar with them.) Personally, I think many companies have long lists of bugs that they do not have time to fix, which might be an interesting angle to speculate about in the context of why bug bounties are so popular.

We totally agree with the reviewer's points, and we admit we took a shortcut in the way that we considered that software testing tools may rather be seen as an exoskeleton tool, expanding human capabilities, rather than a formal way to find bugs. We have rewritten this part, by integrating more deeply with Brady, Anderson and Ball (1999):

"The difficulty of bug hunting is therefore not about finding a bug {\it per se}, but rather about envisioning all possible use situations, which would reveal a software defect (i.e., program crash) or an unintended behavior.

Software solutions have been developed to systematically detect software inconsistencies and thus potential bugs (e.g., Coverity, FindBugs, SLAM, Astree, to name a few). However, 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 explore novel situations -- by manual code inspection or with the help of bug testing software -- in which a software may not behave in the intended way."

2. Fig 1 A: Please redo this plot in a way that is MUCH easier to read the data. There are too many colors jumbled on top of each other, and a casual inspection of the ones I could read doesn't seem to match your 'power law decay over time' claim. e.g., the red bars for Twitter look like they are noisy but not declining over time. In Panel B, your fit would likely look a lot better if you used the cdf (ccdf) instead of the pdf.

Figure 1A: We have unfortunately not found a much better way to represent the timeline, and we have even considered dropping this figure. But the remark by the reviewer stating that Twitter was not really obeying the averaged decay rule, convinced us that the Figure is still informative. We have added a paragraph commenting on the fact that Figure 1B is a statistical average decay, and some programs depart from this average and outperform (resp. underperform) the baseline.

Figure 1B: We inaccurately described the decay as a probability density function, while it is a time series. We have changed our description accordingly

2. Fig. 2, panel B: This is one place where i felt the analysis was flawed. Both of the plots shown on panel B do not look like power laws. Note how the red dots and the green dots are both falling way off the best fit line for large values. This does not look like noise. Also, I question the appropriateness of attempting a power law fit, when your data vary over less than 2 orders of magnitude (x-axis goes from 101 to 10**3).**

We acknowledge the flawed and marginally pertinent nature of Figure 2B, and we have replaced it to present the distribution of bounties per researcher per program: Please see below the reponse provided to Point 3 raised by Reviewer 2.

Concerns raised by Reviewer 1 on the orders of magnitudes have also been incidentally been addressed: the power law distribution of bounties per researcher per program is defined over more than 2 orders of magnitude (on the y-axis because the exponent is larger than 1) and the cut-off is quantified at 400 bounties, which adds to the claim that one expert security researcher cannot scoop a number of bounties per program, which would be dominant or at least large enough to make the statistical moments diverge (stastically and given public data available).

3. Although you appeal to the St. Petersburg Paradox you never actually say what the paradox is. Why is it a paradox? Since you refer to it several times, one sentence explaining it for the non-economics savvy readers would help.

Thanks for noting this issue. We have added a whole detailed paragraph comparing the St. Petersburg paradox with the situation faced by security researchers. Please see below the response provided to Reviewer 2.

4. Quite a bit of the analysis focuses on the decision by individual programmers of whether to switch to a new bounty program when it is first introduced, or to stay with the one they are working on. I was surprised that you did not consider the learning curve involved with becoming familiar with a new software base. I would have expected that researchers might try to amortize their investment in learning about new software by sticking with it, especially as the bounty increases over time.

Reviewer 1 is right to question the obvious existence of a learning curve, which we had not covered explicitly. Our results necessarily embed the learning curve, yet not explicitely and it is obviously hard to measure directly what people have learned and how they have leveraged their acquired knowledge. We have nevertheless added a paragraph in the discussion covering this aspect.

5. Although you cite Ross Anderson's lovely paper from several years ago, I would have appreciated more explanation of how are your results different (or better) than his?

We have expanded the discussion to account for this request by reviewer 1:

Brady, Anderson and Ball (1999) take an evolutionary theory perspective to the problem of software reliability, and basically say that software is sensitive to environment changes (i.e., it is not evolutionary fit) because it is usually designed for one purpose, but then purpose changes over time (think e.g., of software packages in Linux, how they are linked [2], and how they are used in intended ways), and on the contrary to species who adapt by the way of selection (only the fittest portion of the population survives), according to Brady et al., this feature is essentially absent in software.

This distinction between biological and software system may be less relevant nowadays. Software evolution has come to resemble ever more biological systems: While not later than ten years ago, the forking practice was considered as schism between unreconciliable views in a project community (often leading to 2 or more distinct communities with their own new "software breed"), nowadays forking has become a very coming practice on source code hosting platforms (e.g., GitHub). In essence, each forked code repository whether it allows software perform exactly the the same task, or a slightly different task, adds to the stock of code (comparable to the stock of genes) highly desirable for resilience through evolutionary pressure. One may consider two slightly different programs evolved from the same root. At some point, a vulnerability (or a set of vulnerabilities) is found in the most popular one, which are not present in the less popular one. If these vulnerabilities cannot be overcome quickly, and assuming that both programs perform roughly the same tasks, the less popular will end up prevailing. Hence, the more forks, the better overall, event though one may find human cognitive biases (e.g., herding effect [3], competing attention [4]), which don't necessarily exist in nature (we may expand this further discussion in the future).

Here, the focal point is a software piece, or more precisely a set of complementary software pieces, which define the service offered by the focal organization. The fitness of software is assessed by security researchers, may it be internally (internal audit), externally (ethical hackers), or by resorting to the crowd (bug bounty programs). The software runs in a well-defined environment, and it would be hard, if not impossible, to deploy it in other environments (i.e., for a different use, e.g., by a different population), in order to test its robustness. Note that some very large companies by a matter of fact extensively test their software in a variety of environments given the pervasive nature of their service. One may think of Facebook with +1.5 billion users across the world.

Because they all carry their own unique experience, security researchers offer a form of confrontation with alternative environments. Additionally, the more remote from the organization actually writing or sourcing the code, the more original the view on the software piece (without the hassle of deployment). This is where our paper offers a similar conceptual view with the one of Brady, Anderson and Ball (1999), and with the quote by Eric Raymond "Given enough eyeballs, all bugs are shallow". In sum, diversity of views prevails over accumulated expertise, although we make no claim that expertise is not required. We just observe that its effects are just bounded.

A slightly shorter version of the above argument has been incorporated in the discussion:

(latex source)

This observation is reminiscent of an early proposition on the topic: Brady et al. \cite{brady1999murphy} took an evolutionary theory perspective to the problem of software reliability, and basically said that software is sensitive to environmental changes (i.e., it is not evolutionary fit) because it is usually designed for one purpose. The purpose however changes over time (think e.g., of software packages in Linux, how they are surprisingly linked together \cite{maillart2008empirical}, and how they are used in unintended ways). On the contrary to species who adapt by the way of selection (only the fittest portion of the population survives), this feature is essentially absent in software, according to Brady et al.
%The distinction between biological and software system may be less relevant nowadays. Software evolution has come to resemble ever more biological systems: While not later than ten years ago, the forking practice was considered as a schism between unreconcilable views in a project community, often leading to 2 or more distinct communities with their own new software breed, nowadays forking has become a very coming practice on source code hosting platforms (e.g., GitHub). In essence, each forked code repository whether it allows software perform exactly the same task, or a slightly different task, adds to the stock of code (comparable to the stock of genes) highly desirable for resilience through evolutionary pressure. One may consider two slightly different programs evolved from the same root. At some point, a vulnerability (or a set of vulnerabilities) is found in the most popular one, which are not present in the less popular one. If these vulnerabilities cannot be overcome quickly, and assuming that both programs perform roughly the same tasks, the less popular will end up prevailing. Hence, the more forks, the better overall, event though one may find human cognitive biases (e.g., herding effect \cite{salganik2006experimental}, competing attention \cite{hansen2001competing}), which don't necessarily exist in nature.
Here, the focal point is a software piece, or more precisely a set of complementary software pieces, which define the service offered by the focal organization. Software fitness is assessed by security researchers, internally (internal audit), externally (ethical hackers), or by resorting to the crowd (bug bounty programs). The software runs in a well-defined environment, and it would be hard, if not impossible, to deploy it in other environments (i.e., for a different use, e.g., by a different population), in order to test its robustness. Note that some very large companies by a matter of fact extensively test their software in a variety of environments given the pervasive nature of their service. One may think of Facebook with more 1.5 billion users worldwide.

Because they all carry their own unique experience, security researchers offer a form of confrontation with alternative environments. Additionally, the more remote from the focal organization, the more original the view on the software piece (without the hassle of deployment). Our results offer a similar conceptual view, as well as with the quote by Eric Raymond "Given enough eyeballs, all bugs are shallow". In sum, diversity of views prevails over accumulated expertise, although we make no claim that expertise is not required. We just observe that its individual effects are just bounded. These results also cast questions on the learning curves, and incentives to keep digging bugs in a program, which the researcher is already familiar with. We observe that overall these incentives become quickly insufficient in comparison with the increasing difficulty for a researcher to find additional bugs.

Many small writing issues. Here is a representative (but not exhaustive) sample:

"In a nutshell, the proposed (monopsonistic) auction mechanism implies an initial R(t = t0) = R0, which increases linearly with time. If a vulnerability is reported more than once, only the first reporter receives the reward." Unclear what the equation means at this point in the text. And how does the following sentence relate to this?

An important word was missing (i.e., "reward"), the sentence has been correctd accordingly: *In a nutshell, the proposed (monopsonistic) auction mechanism implies an initial reward R(t = t0) = R0, which increases linearly with time.*

"but it generally requires a high level of programming proficiency coupled with hacking skills to piece things apart, in order to understand them better. " This reviewer would 'piece things together' not apart.

The formulation was confusing and therefore suppressed:*"but it generally requires a high level of programming proficiency coupled with hacking skills, in order to understand them better."*

'to get security researchers hunt bugs' (missing 'to')

Thank you! The missing word was added

"α = −0.40(4) (p < 0.001 and R2 = 0.79)"
(4) is confusing.
"α = 1.10(3)"
(3) is confusing

Even though this may be confusing to the Reviewer, this is a canonical, yet admittedly relatively uncommon, notation for the standard error of the estimate.

'Security researcher enrollment is determinant for the success of a bug bounty program' ==> ... enrollment determines success ...

Thanks for offering a much more elegant word arrangement. We have rephrased the title as: "Security researcher enrollment determines the success of a bug bounty program"

"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." (ungrammatical)

We have replaced this sentence by a much simpler one: *"For security researchers, the main metric is the expected cumulative payoff earned from the accumulation of bounty awards."*

References:

[1] Wheatley, Spencer, Thomas Maillart, and Didier Sornette. "The extreme risk of personal data breaches and the erosion of privacy." The European Physical Journal B 89.1 (2016): 1-12.
[2] Maillart, T., et al. "Empirical tests of Zipf's law mechanism in open source Linux distribution." *Physical Review Letters* 101.21 (2008): 218701.
[3] Salganik, Matthew J., Peter Sheridan Dodds, and Duncan J. Watts. "Experimental study of inequality and unpredictability in an artificial cultural market." *science* 311.5762 (2006): 854-856.
[4] Hansen, Morten T., and Martine R. Haas. "Competing for attention in knowledge markets: Electronic document dissemination in a management consulting company." Administrative Science Quarterly 46.1 (2001): 1-28.

Reviewer 2

Overall, I enjoyed reading this submission. I believe the WEIS community would be very interested in this topic and seeing the paper presented. Even spending time discussing the data collected from HackerOne would probably be informative. Some comments:

1. The paper makes several references to the St. Petersburg paradox, but it wasn't clear to me why. From what I understood from the paper's findings, the probability decreases faster than the rewards scale. Thus, expected payoff does not diverge in contrast to that paradox. On that dimension, I didn't fully understand the statement: "We observe that behaviors seem to contradict rational behavior dictated by the expected utility function, characterized by a structure comparable to the St. Petersburg." Perhaps the authors could clarify the point they are trying to make here.

We agree that we have not been clear when referring to the St. Petersburg paradox, and we have at first considered dropping all references to this phenomenon. Instead,

we have made all effort to integrate it better to the rest of the text, in the following ways (latex source):

The St. Petersburg paradox states the problem of decision-making when both the probability and the reward are diverging when $k \rightarrow \infty$: A player has a chance to toss a fair coin at each stage of the game. The pot starts at 2 and is doubled every time a head appears. The first time a tail appears, the game ends and the player wins whatever is in the pot. Thus the player wins 2 if a tail appears on the first toss, 4 if a head appears on the first toss and a tail on the second, 8 if a head appears on the first two tosses and a tail on the third, and so on. The main interest of Bernoulli was to determine how much a player would be ready to pay this game, and he found that very few people would like to play this game even though the expected utility increases (in the simplest case proposed by Bernoulli, $U_n = \sum_{k=0}^{n} U_k = n$) \cite{bernoulli1954exposition}. For bug bounty programs, the situation is slightly similar because the main question a security researcher may ask herself when enrolling a bug bounty program is the amount of initial effort (i.e., the upfront costs) to be devoted in order to make a positive expected net payoff. This payoff itself conditioned by the expected number of bug discoveries and their associated monetary rewards minus the cost. The situation of a security researcher differs from the St Petersburg lottery, as bug search costs are incurred at every step. Since these costs influence the probability to find an additional bug, for the sake of simplicity, we assume that they are integrated in $P(k)$. The security researcher may also decide to stop searching for bugs in a program, at any time. This is equivalent to setting $P(k+1) = 0$.

2. One thing I would have liked to see discussed more is how bug rewards programs "should be" designed in light of the paper's findings. In that higher ranked bugs become costlier to find and necessitate higher rewards, program managers likely do not want to maximize the total number of bugs discovered as is suggested at the beginning of Section 5. Perhaps costly bugs to discover should never be found - this seems like an economically efficient outcome.

Reviewer 2 is pointing to interesting discussion, which we would have indeed not envisioned: Some bug may be so costly to discover that it is just not worth the effort, assuming that no attacker is likely to face the same difficulty. Our results tend to show that this intuition, which we understand to be well-grounded in the common representative agent assumption in economics, is inaccurate here because the accumulation of bug discoveries is the result of a large crowd of white hats (or at least grey hats) finding and reporting each an average number of bugs.

If we consider an arbitrary focal bug to be discovered, the chance that it will be discovered increases with the number of researchers. If half security researchers interested in the security of the focal software are black hats, there is roughly 50% chance that the focal bug will be discovered by a black hat. If the proportion of population types (white and black hats) compounded over time is uneven, then the probability of discovery falling in one of both categories changes accordingly.

(latex source)

Moreover, we find that the larger the population of enrolled researchers, the even more bugs are found. In that process, the initial windfall effect of a newly launched program is critical and determines an important portion of the bug discovery timeline, and accordingly researchers are ready to switch their attention towards newly launched programs, at the expense of older ones. These results have critical implications for software security: If we consider an arbitrary focal bug to be discovered, the chance that it will be discovered increases with the number of researchers. If half researchers interested in the security of the focal software are black hats, there is roughly 50\% chance that the focal bug will be discovered by a black hat. If the proportion of population types (white and black hats) compounded over time is uneven, then the probability of discovery falling in one of both categories changes accordingly. In other words, in order to be effective (statistically speaking), a bug bounty program must reach much more white hats, compared to the estimated amount of black hats interested in finding holes in the focal software.

3. There are some concerns with fit in Figure 2B. The curvature seen in the data in this log-log plot seem to suggest a different relationship than might be adequately captured by the functional specification.

We totally agree with Reviewer 2 (and Reviewer 1 as well): Not only Figure 2B was less than convincing, but it also brought very little additional information to support the claim that the number of bounties found in each program is mostly driven by the "number of eyeballs". We have thus replaced Figure 2B by a fit of the tail distribution of bounties found per program per researcher, which is informative of the relative influence of high performers versus average performers. We have thus added the figure and appended the caption as follows (latex source):

{\bf B.} The tail distribution of bounty discoveries per researcher per program follows a power law distribution $P(X>x) \sim 1/x^\gamma$ with $1 < \gamma = 1.63(7) < 2$. The distribution is therefore relatively well bounded (with the first moment being well-defined). Furthermore, we observe an upper cut-off of the tail with $x_{max} \approx 400$ bounties. Thus, from {\bf A.} and {\bf B.} combined, we find that the number of vulnerabilities is mainly driven by the number of researchers enrolled in programs.

Accordingly, we have also adapted the section entitled*: Security researcher enrollment determines the success of a bug bounty program*

4. Section 5.3 would benefit from further development.

We have slightly refined Section 5.3, but in our opinion, there is no much more to be added, and since Reviewer 2 did not provide further detailed requests, it is hard to read her mind. In a nutshell, this section further verifies that indeed:

(i) programs launched by more prominent organizations (measured by their Alexa rank) receive more attention, more security researcher enrollment, and *in fine* more valid bugs

(ii) New programs change researcher incentives in a detrimental way to older programs.

Both aspects can be inferred from previous results, but we wanted to make these important points further clear and accessible to the reader. We believe using a simple, yet not simplified, OLS model does a good job. For now, it is not our intention to develop a more sophisticated econometric model, which would e.g., test how the organization design specificities of bug bounty programs may further re-inforce (or on the contrary dampen) incentive changes. This certainly could be part of some future work.

Reviewer 3:

The authors studied 35 bug bounty programs run by HackerOne and found an initial windfall followed by a power law decay with exponent about 0.4. More surprising is that bug finding productivity increases with the number of bug hunters; the authors suggest this is because the more researchers there are, the more likely you are to get a good one (of which I'm sceptical; does this give a power law with expeonent 1.1?0 and separately that it's about cross-fertilisation – bug hunters build on each others' work. It might also be a competititon effect.

We are thankful to Reviewer 3 for sharing her thoughts. As we have realized following the reviews, the pooling effect was not completely accurately described. In the reviewed submission, we inferred that some researchers dominate bug discoveries (in a given program). We have refined our study, which led us to replace Figure 2B, which shows a slightly different picture: Each researcher enrolled in a program tends to make a significant contribution of bug discoveries, and those who perform best, do not perform wildly better compared to the average.

We have accordingly refined the way the article is presented, to stress further that the diversity of security researchers (best represented by their number and the fact that each individual conribution is bounded), is key for the success of a bug bounty program.

The fact that bug hunters switch from one program to another probably also has to do with the fact that different hunters find different bugs, as their skills and tools lead them to search different parts of the problem space.

Intuition tells us a similar tale, but former research has not found strong evidence [5] and we have not either. Our interpretation has been so far that there are too many conflicting incentives driving researcher decisions besides the pure technical / skill / learning aspects. And thus the latter have remained merely unobservable, even though they most surely play a role.

The hypotheses you test are basically testing the finiteness of the pool of bug hunters, or perhaps more formally the price elasticity of supply of bug hunting. It turns out that there is indeed a strong displacement effect; there are not that many eyeballs, and only a few of them belong to stars. In short, bug bounty programs work, but they have their limits, set by competition for first-class bug hunters.

To rebound on our response provided to Reviewer 3 above (as well as Reviewers 1 and 2 before), we think that the success of a bug bounty program is essentially driven by the capacity of the management to attract a large number of security researchers, with diverse background and knowledge skills.

References:

[5] Zhao, Mingyi, Jens Grossklags, and Kai Chen. "An exploratory study of white hat behaviors in a web vulnerability disclosure program." Proceedings of the 2014 ACM Workshop on Security Information Workers. ACM, 2014.