Economics Bulletin

Volume 38, Issue 1

Economic Motivations for Software Bug Bounties

Christopher Sprague
Rochester Institute of Technology

Jeffrey Wagner
Rochester Institute of Technology

Abstract

Some software developers sponsor bug bounty programs, whereby outside parties with comparatively lower costs are compensated for finding bugs. We propose a basic model of why some developers offer bounties while others don't, and why those that do offer bounties typically outsource only a portion of the bug-finding. Our relatively basic framework and preliminary result can support further investigation of public policy instruments, such as products liability law, aimed at modulating software failures that may have large public impacts.

We appreciate several helpful comments and suggestions from session participants at RIT's 10th Annual Economics and Public Policy Student Research Conference. This work began as the first author's undergraduate capstone project while double-majoring in Computer Science and Economics.

Citation: Christopher Sprague and Jeffrey Wagner, (2018) "Economic Motivations for Software Bug Bounties", *Economics Bulletin*, Volume 38, Issue 1, pages 550-557

Contact: Christopher Sprague - css7209@rit.edu, Jeffrey Wagner - mjwgse@rit.edu.

Submitted: December 31, 2017. **Published:** March 23, 2018.

1. Introduction and Literature Review

The purpose of this note is to set forth a basic economic model of software bug bounties. Such bounties are an emerging trend in software markets, wherein software developers offer to pay professionals outside the firm (a.k.a., bug hunters) for finding bugs that slip past the firm's own care in finding them. Not all software companies offer bug bounty programs; among those that do, some bug bounty programs are relatively substantial. For instance, Facebook's bug bounty program has paid $5 million in roughly the past five years[1]; several additional examples are noted by Hunt (2017). Both software developers and public policy professionals seeking to maximize social welfare from software are interested in bug bounty programs for several reasons. First, all parties are interested in finding bugs at the lowest marginal cost, and where the marginal benefits and marginal costs of finding bugs are equal. Second, finding more bugs raises software quality, which raises demand for software. Third, a bug bounty program can reduce not only the firm's costs, but consumer prices as well—both consumer and producer surplus can rise via an effective bug bounty program. Fourth, bug bounty programs enable external bug reporting and discourage "black market" malicious bug knowledge activity. The main contribution of our paper is to combine elements of the existing literature to motivate a basic economic model of bug bounties. That is, while multiple papers touch upon the topic of bug bounties in the course of focusing upon other aspects of software, our approach is to feature bug bounties at the core of the model.

Before setting forth our model in the next section, we survey the relatively short existing literature that comments upon the strengths and weaknesses of bug bounties. Barnes (2004, 322-24) was an early commentator in the legal scholarship on the topic of bug bounties within the more general topic of whether or not software developers should be held liable for software failures. He notes that it is not immediately clear why some firms have bounty programs and others do not, and that perhaps mandatory bug bounty programs should be established. Hahn and Layne-Ferrar (2006, 338), citing Barnes (2004), also explore several strengths and weaknesses of holding developers responsible for software security problems. Within that analysis, they suggest that bug bounties have been moderately successful and cost-effective for modulating some software issues. Moore (2010, 107-108, 113) describes the challenges and opportunities in utilizing either ex ante care standards or ex post liability for dealing with software security issues, and the vulnerability faced by most consumers who are typically limited to installing patches and maintaining their anti-virus software as defense against bugs. Choi *et al.* (2010, 885) provide a model in which voluntary and mandatory security vulnerability disclosure policies can be compared. Within their disclosure model, they show that bug bounty programs can be welfare-improving. Lam (2016, 49) emphasizes that software developers invest effort in both attack prevention and damage control (i.e. multidimensional care), and that implementing bug bounty programs can have the undesirable effect of relaxing effort directed to attach prevention in favor of hoping to deal with any software issues down the road instead. So instead of promoting bug bounties, he advocates the joint use of a standard of care with a partial liability rule.

Most recently, Kesan and Hayes (2016), citing Choi *et al.* (2010), propose building an exchange market for software security vulnerability information that falls somewhere between

[1] https://www.facebook.com/bugbounty, accessed 12/12/2017.

the black market for information and the so-called white market for information where traditional bug bounty programs reside. That is, their concern is for incentivizing the exchange of information related to the most serious security vulnerabilities that, if not traded in grey or white markets, will transact in black markets instead. Kesan and Hayes (2016, 760-1) note that typical bug bounties do not offer compensation that is competitive with what researchers can earn if they sell the information to someone else. So their focus is upon the design of exchanges wherein third-parties (perhaps the government) would receive claims of found severe vulnerabilities from hunters; evaluate and score those claims according to a threat severity index; present software developers with the threat severity index; and help the parties negotiate a price. A third-party mechanism is arguably crucial to the mechanism because of the peculiarity of information as a good. If the hunter reveals the specifics of the vulnerability directly to the software developer, the developer will learn what it wants to know and will have no incentive to pay for it. Likewise, developers do not have an incentive to pay for the information unless they can evaluate the quality of the information ex ante. Some type of information must be revealed in order for a price to materialize, and a third-party broker can facilitate that.

To summarize our review of the literature, we find that each of these papers features one or more elements of the law, economics, and technology that needs to be taken into account in order for further progress to occur on this important topic. As noted at the outset, our goal is to propose a relatively simple model that combines elements from each of the above references to motivate bug bounties per se and that can support analysis in future research of how various policy instruments might be used singularly or jointly to effect Pareto-improvements. We present the basic model in Section 2 and we conclude with some directions for future research in Section 3.

2. The Baseline Models of Social and Private Optimality

We begin by setting forth notation for our baseline model of social and privately optimal software bug-finding. As is standard in the law-and-economics literature, we model the decision variable as care—in this particular context, the amount of care software developers and external bug hunters should each expend finding bugs in a software program. Suppose damage from software bugs occurs at an expected dollar rate $D(x) = p(x)H$ with D continuous and twice differentiable in bug-finding care x, and $\frac{dD}{dx} < 0$, $\frac{d^2D}{dx^2} \geq 0$. The "expected" aspect of the damage function conveys the fairly realistic assumption that software developers and/or bug hunters sample code for bugs; there is a probability p between zero and one that a bug will be found upon sampling, and when found, we assume there is an average amount of harm H done if the bug is not fixed (and that would also occur if the bug is not found to begin with). For simplicity, we let H be constant—that is, independent of x. Harm H could vary with x, just as p varies with x, but this merely complicates the algebra without generating additional insights from the model. We assume throughout the paper that decision-makers are risk-neutral, such that they care about outcomes in expected value only. We also assume that software consumers are passive, i.e., that they are not able to take actions to detect bugs.[2] Our assumptions on the expected damage

[2] This is arguably a fairly realistic assumption in that evidence shows consumers tend not to be very mindful of installing developer-provided patches; maintaining nor running anti-virus software; or trouble-shooting software glitches. And software developers are increasingly able to push updates/patches to consumers automatically

function *D* say that the marginal expected damage from bugs that is avoided by taking care is positive but diminishing as bug-finding care increases. For simplicity, we assume that the third derivative of the damage function is zero. Hence, the marginal damage avoided function could be a horizontal or downward-sloping linear function.

While bugs cause expected damage *D*, software developers and other professionals such as bug hunters face costs in taking care to find and fix bugs. Suppose the cost of care involved in identifying and fixing bugs is given by $C(x)$, which is continuous and twice differentiable, increases as *x* increases, and is the horizontal summation of all individual cost functions in the economy (i.e. of care taken by private software developers as well as bug hunters). As in Lam (2016), we set aside the demand side of the firm's and society's concern for now and focus just upon costs. The social planner's problem is to have care taken so as to minimize the social costs of software bugs, where there are costs *D* of having bugs and costs of care *C* to reduce bugs. The social optimization is thus:

$$min\ Z_s(x) = D(x) + C(x) \tag{1}$$

Taking the first-order condition, we have

$$\frac{dZ_s}{dx} = \frac{dD}{dx} + \frac{dC}{dx} = 0 \tag{2}$$

Eq. (2) leads to the choice of x^* where $-\frac{dD}{dx} = \frac{dC}{dx}$, noting that $\frac{dD}{dx} < 0$.

In contrast, the private software developer cannot be assumed to take the full social damage from bugs into account. Rather, the developer only has incentive to take into account a private subset of *D*, say $D_p < D$, which for simplicity can be normalized to zero. The private subset of *D* would include the firm's financial cost to restore computing services affected by undetected bugs and harm to its reputation among users aware of persistent bugs and their effects. However, since indirect users of the buggy software may also be affected by software failures, and since some users may not be aware of persistent bugs and their effects, and since software developers are currently shielded from products liability, the private firm's assessment of expected damages is almost sure to be less than society's assessment. In that case, the software developer's private optimization can be modeled as:

$$min\ Z_p(x) = C(x) \tag{3}$$

Taking the first-order condition, we have

$$\frac{dZ_p}{dx} = \frac{dC}{dx} = 0 \tag{4}$$

The private developer will choose care $\bar{x} = 0$ in this abstract case, corresponding to the origin in Figure 1. What eq. (4) tells us is that the private firm in this abstract case will optimally choose not to take care to find bugs; firms will rather favor promoting a *caveat emptor* software market.[3]

---

through the internet. Thus we normalize consumer actions to zero and focus upon the decisions software developers and other professionals such as bug bounty hunters are able to take that abate bugs.

[3] Again, note that we normalized the private damage the software developer sustains from the persistence of bugs to be zero for simplicity. As per Figure 1, if there does not exist an *MD* function, then it is optimal to minimize just

This inclination not to take socially optimal care will not generate an infinite number of bugs; after all, software firms are not in the business of making bugs. Rather, the software developer has an incentive to let stand a finite but possibly large number of bugs rather than to expend care resources abating them. Equations (2) and (4) together imply that there is room for public policy to incentivize private market care in detecting and fixing software bugs. That is, policy makers are concerned to increase bug-reducing care from a rate of $\bar{x} = 0$ to a rate of $x^* > 0$. The baseline model is illustrated in Figure 1 below, where triangle *ab0* demarks the social surplus to be captured by successful policy that increases care *x* taken to find and resolve software bugs.

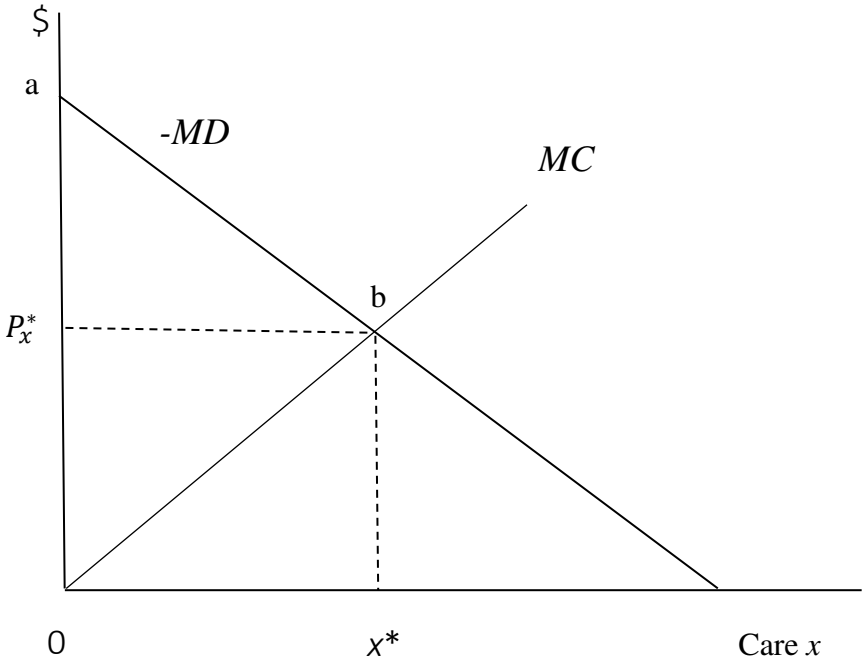

Figure 1: Comparing Privately and Socially Optimal Care *x*

As others have noted, there are in principle several policy approaches one can take to create Pareto-improvements in this situation. These options include the imposition of liability on software developers for harm done by buggy software. Each of those policies has strengths and weaknesses. What we focus upon here is the role of bug bounty programs, wherein, according to the equi-marginal principle, software developers could outsource bug finding such that the marginal costs of care of developers and bug hunters are equalized and social costs of software are minimized. Figure 2 illustrates in a dual-*x*-axis framework the standard case in which developers do have incentive to outsource some but not all bug hunting activity, and therefore to have some interest in bug bounty programs. In the illustrated case, $x_d^* + x_h^* = x^*$ and both parties take positive rates of care *x*. This would be considered a voluntary step on the

---

with respect to the cost of care; since care is costly, the cost is minimized by choosing zero care. In the real world, the software developer probably spends a relatively modest amount of effort finding bugs in this abstract case of not being accountable for any damage from bugs, and suffering any financial consequences. We've also proceeded with the assumption that the marginal cost function illustrated in Figure 1 is linear and begins at the origin. This function could certainly have a positive second derivative, and it could have a positive y-axis intercept. Relaxing either of these assumptions would not alter the baseline result in which the private firm chooses less care than is socially optimal.

developer's part to ensure that greater care is taken to find software bugs. Two corner solutions are possible, however. At one extreme case, if the MC_h for hunters is everywhere above the MC_d for developers, such that the *MC* functions in Figure 2 do not intersect in the positive orthant, all bug hunting will be carried out in-house and the developer will not voluntarily promote a bug bounty program. Hence, our model provides a simple motivation and visual for the observation that many developers do not have bug bounty programs. Likewise, it is theoretically possible for the MC_d of developers in Figure 2 to be everywhere above (and hence not intersect) the MC_h of hunters, in which case all bug hunting would be outsourced from developers to hunters and such developers would be quite reliant upon successful bug bounty programs. However, we would not expect to see such a scenario in practice: a software firm that outsources 100% of bug finding would be rather unusual.

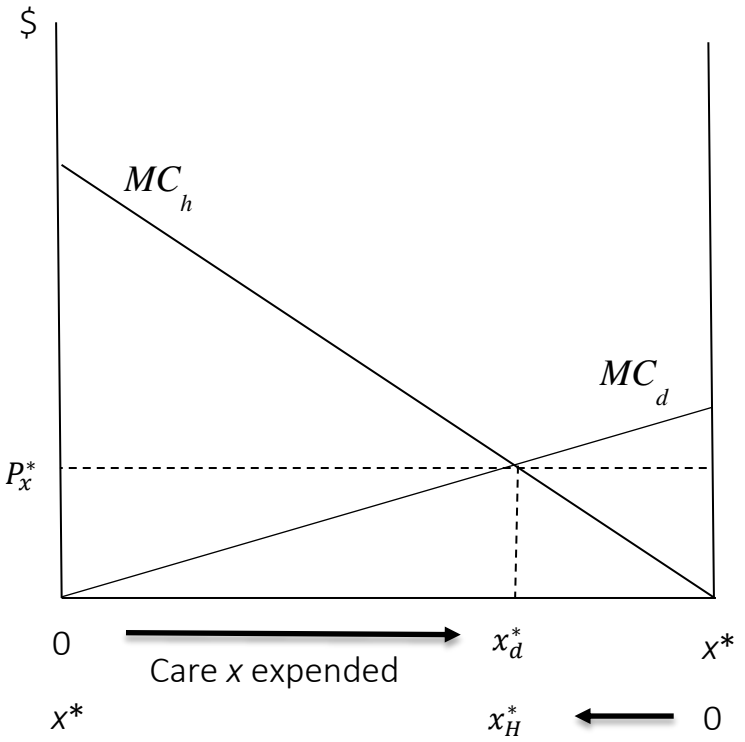

Figure 2: Division of Care by Equi-Marginal Principle

If a developer begins at x^*, at the far right of Figure 2, wherein he or she assumes all responsibility for finding bugs and bug hunters expend no effort finding bugs, we can immediately see the developer's incentive to offer a bug bounty. Taking a one-unit step to the left, we see that the developer's *MC* is far above the hunter's *MC*. The vertical distance between *MC* functions demarks the range of Pareto-improving bounties. For a simple example, MC_d could be \$500 and MC_h could be \$100 at $x_h = 1$. The idea of the bug bounty is that both the developer and hunter will be better off if the developer agrees to pay the hunter any amount between \$100 and \$500 for the hunter's first unit of bug-hunting effort. Such Pareto-improving trades exist for all x as we move right to left along the x-axis until $MC_d = MC_h$. At that point, the developer is indifferent to expending the next unit of bug-finding care with her in-house resources or outsourcing that unit of care to the professional bug-hunting market. Thus, bug

bounty programs work in the same manner that competitive, tradable emission permit markets work to minimize society's cost to reach any particular environmental quality level.

3. Conclusions and Directions for Future Research

The objective for this paper was to set forth a basic model of the economics of bug bounty programs. Using a standard equi-marginal cost comparison framework, we are able to see how differences in agents' marginal cost of taking care to find bugs create opportunities for Pareto-improving trades in the form of software developers offering bug bounties to bug hunters. The gains-from-trade model enables us to likewise derive and visualize graphically the boundary conditions under which developers would choose not to offer a bug bounty program or to completely outsource the search for bugs to bug hunters. While others in the literature discuss the strengths and weaknesses of bug bounty programs—and the policy choices of imposing mandatory bug bounty programs as opposed to encouraging developers to utilize them voluntarily—we are not aware of previous papers that illustrate this basic gains-from-trade motivation for voluntary engagement with bug bounty programs.

While the framework is basic and based upon well-known concepts, its purpose is to provide a general structure in which alternative policy instruments that promote Pareto-improvements in care can be analyzed. One key area for extension regards how bug bounty programs interact with—indeed, may be motivated by—the credible but low-level threat of liability for harm that software developers face. That is, legal commentators such has Scott (2008) have suggested that while software developers currently enjoy broad legislative insulation from products liability claims for software failures, there may come a time when large-scale software failures motivate the public (through the courts and through its elected legislatures) to revisit this insulation. Such a possibility calls to mind Kahneman *et al.* (1986), who warn firms to take heed of currently legal actions that could nevertheless provoke costly public policy reversals. In the environmental economics literature, scholars such as Segerson and Wu (2006) and Brouhle *et al.* (2009) show that complementing a voluntary pollution prevention approach with a credible threat to impose more stringent regulations in industries where voluntary programs fall short of social expectations can motive Pareto-improvements. Likewise, in our software bug context, we believe that appropriate extension of our basic model would show that the low but positive probability of being found liable for harm from software weaknesses efficiently complements (and to some degree motivates) bug bounty programs that move the privately optimal care to find bugs closer to socially optimal levels.

Such an extension that emphasizes the uncertain threat of liability, as opposed to the actual imposition of liability, on developers complements Lam's (2016) insightful analysis. The difference is that in Lam's model, he assumes that a policy of partial liability could be imposed with certainty (i.e., that the probability of conviction is one), and that the standard of care that Lam suggests should be imposed jointly is also known with certainty by all parties. Our review of the literature suggests these assumptions may be sufficiently strong that we should also consider the case in which developers are not certain of how much care is necessary to preserve legal immunity from liability for software quality issues. Developers are likely also quite uncertain how much care is necessary to meet an ambiguous care standard. Elements of Bhole and Wagner's (2008) analysis of taking optimal multidimensional (observable and unobservable) care in the presence of uncertain conviction for harm could be useful in our framework. Specifically, the software developers' care could be considered unobservable care, while

developers' expenditures on bug bounties could be considered observable care. Suppose that changes in legislation that shield software developers from products liability are sensitive to frequent occurrences of major security flaws that are modulated by care, and that changes in legislation are also sensitive to legislators' observations of developers' care. If developers' private care is difficult to observe, then developers' investments in observable care in the form of bug bounties yield both reductions in expected damages *as well as* reductions in the likelihood of liability exposure via policy reversal. We hypothesize that this hedge against liability policy reversal may in fact be the largest benefit of bug bounty programs to software developers.

Second, our model can be generalized to investigate how the introduction of a bug bounty program in the presence of potential liability affects R & D investment and innovation in subsequent periods. That is, to our knowledge, the literature on bug bounties focuses upon their impact within static models. However, more general economic literature shows that policy levers that have desirable properties in a one-period model may have different properties in multi-period models. For instance, Endres and Bertram (2006) describe how the development of care technology in a dynamic model is affected by different liability rules. Elements of their framework could be added to ours to investigate how bug bounties, with and without potential liability, affects firms' incentives to invest in its own care technology. Since innovation has a public-good component, we are almost sure to find that private decision-making will lead to suboptimal social care technology investments. The resulting model would shed light on the best ways forward in that case.

4. References

Barnes, D. A. (2004) "Deworming the internet" *Texas Law Review* **83**, 279-329.

Bhole, B. and J. Wagner (2008) "The joint use of regulation and strict liability with multidimensional care and uncertain conviction" *International Review of Law and Economics* **28**, 123-132.

Brouhle, K., Griffiths, C. and A. Wolverton (2009) "Evaluating the role of EPA policy levers: An examination of a voluntary program and regulatory threat in the metal-finishing industry" *Journal of Environmental Economics and Management* **57(2)**, 166-181.

Callan, S. J. and J. M. Thomas (2013) *Environmental Economics and Management: Theory, Policy, and Applications*, Sixth Edition, Cengage: Boston.

Choi, J. P., Fershtman, C. and N. Gandal (2010). "Network security: Vulnerability and disclosure policy" *The Journal of Industrial Economics* **LVIII (4)**, 868-894.

Endres, A. and R. Bertram (2006) "The development of care technology under liability law" *International Review of Law and Economics* **26**, 503-518.

Hahn, L. W. and A. Layne-Ferrar (2006) "The law and economics of software security" *Harvard Journal of Law and Public Policy* **30**, 283-353.

Hunt, T. (2017). https://www.troyhunt.com/fixing-data-breaches-part-4-bug-bounties/. Accessed December 20, 2017.

Kahneman D., Knetsch, J. L. and R. Thaler (1986) "Fairness as a constraint on profit seeking: Entitlements in the market" *American Economic Review* **76(4)**, 728-741.

Kesan, J. P. and C. M. Hayes (2016) "Bugs in the market: Creating a legitimate, transparent, and vendor-focused market for software vulnerabilities" *Arizona Law Review* **58**, 753-830.

Kolstad, C. D. (2011) *Environmental Economics*, Second Edition, Oxford UP: Oxford, UK.

Lam, W. M. W. (2016) "Attack-prevention and damage-control investments in cybersecurity" *Information Economics and Policy* **37**, 42-51.

Moore, T. (2010) "The economics of cybersecurity: Principles and policy options" *International Journal of Critical Infrastructure Protection* **3**, 103-117.

Scott, M. D. (2008) "Tort liability for vendors of insecure software: Has the time finally come?" *Maryland Law Review* **67(2)**, 425-484.

Segerson, K. and J. Wu (2006) "Nonpoint pollution control: Inducing first-best outcomes through the use of threats" *Journal of Environmental Economics and Management* **51(2)**, 165-184.