An Economic Analysis of Third-Party Software Reliability Improvement using the Bug Bounty Program

Tianlu Zhou
Tianjin University, imis_ztl@tju.edu.cn

DAN MA
Singapore Management University, madan@smu.edu.sg

Nan Feng
Tianjin University, fengnan@tju.edu.cn

An Economic Analysis of Third-Party Software Reliability Improvement using the Bug Bounty Program

Completed Research Paper

Tianlu Zhou
Tianjin University
Tianjin, China
imis_ztl@tju.edu.cn

Dan Ma
Singapore Management University
Singapore
madan@smu.edu.sg

Nan Feng
Tianjin University
Tianjin, China
fengnan@tju.edu.cn

Abstract

Bug Bounty Programs (BBPs) reward external hackers for reporting software vulnerabilities. As the number of security issues caused by third-party applications has been significantly increased recently, many digital platforms are considering launching BBPs to help improve the reliability of third-party software. In this paper, we present an analytical model to examine the strategic decisions of launching and participating in a BBP for the platform and the third-party vendor, respectively. We find that the platform's (the vendor's) BBP launching (participation) decisions depend on two key factors: the expected loss due to security breaches and the vendor's reliability investment efficiency. We show that the incentive of using BBP, for the platform and vendor, sometimes is inconsistent. Meanwhile, we find that using the BBP is not always socially optimal. Under certain conditions, it reduces the overall software reliability, instead of improving it, makes the platform marketplace less secure, and thus hurts end users.

Keywords: bug bounty program, third-party software, digital platform, software reliability

Introduction

Nowadays, third-party software plays a critical role on digital platforms. For example, the Apple Store contains approximately 2 million pieces of software, with only about 60 of them developed by Apple itself while more than 99.99% being third-party applications (Caminade & Wartburg, 2022). Third-party software adds significant value to digital platforms. Their amount and services keep expanding, so are the breaches and losses caused due to their vulnerability. According to Gartner (O'driscoll, 2023), cybercriminals are increasingly using third parties to attack crucial targets. For instance, because of vulnerabilities in third-party software, 540 million records on Facebook were exposed in 2019 (Lapowsky, 2019).

Digital platforms, therefore, are actively looking for ways to improve the reliability of third-party software to mitigate security breach loss. Many platforms have launched Bug Bounty Programs (BBPs) for third-parties: Platforms offer financial rewards to external ethical hackers for testing third-party software and reporting valid vulnerabilities. Facebook, for example, pays hackers for reporting security bugs in third-party software (Kumar, 2019). Google, similarly, throws bug bounty bucks at mega-popular third-party applications in its Play Store (Vaas, 2019).

Platforms are expecting several benefits of using BBPs. First, under BBPs, external hackers are motivated to legally report bugs and vulnerabilities, instead of using them to launch attacks maliciously, leading to threat reduction (Zhou & Hui, 2021). In addition, with BBPs, software vendors will verify and fix valid bugs reported by external hackers and thus further improve the reliability of their software, which makes the platform a more secure marketplace and thus obtain higher profits.

Not all platforms, however, launch BBPs for third-parties. TikTok, a leading social media platform, is one example. In its bug bounty guidelines, it stipulates that ethical hackers' research should not involve applications, products, or services of a third party (TikTok, 2023).For the platform, there are also some concerns about launching BBPs. One obvious reason is the costs associated with BBPs that the platform must pay bug bounty rewards. Besides, the platform also needs to take the third-party vendors' reaction into considerations. Knowing there will be ethical hackers in the later stage who help to discover vulnerability of their products, will the third-party vendor choose to reduce the initial investment in the software reliability investment? If so, what will be the ultimate software reliability level? This is an unclear issue. Thus, the platform must find out under what conditions it is profitable to launch a BPP for the third-party software and understand how would its launching decision impact the vendor's behavior. This is the first research question we will address in this work.

Only being authorized and verified by the third-party vendor, external hackers can earn the rewards provided by the platform (Google, 2023; Facebook, 2022). The third-party vendor, however, could choose not to participate in BBPs launched by the platform if it finds not worthwhile to do so. It must evaluate and tradeoff the expected benefits and costs. On the positive side, if the vendor participates in the BBP, it enjoys the threat reduction and potential revenue increase as the platform does. Meanwhile, the vendor incurs BBP related costs of processing bug reports and fixing valid bugs raised by external hackers. In practice, it has been turned out that the rate of valid bug reports is quite low, suggesting that the vendor ends up paying great efforts for identifying invalid bug reports. For instance, in 2022, Facebook estimates that the validity rate of bug reports of its BBP was only 7.5% (Oren, 2022). In addition, with the help of external hackers, the vendor shall adjust its software reliability investment level accordingly. Hence, the second research question of our work is related to the software vendor's optimal decisions: When should the third-party vendor participate in a BBP launched by the platform? And if so, what is the vendor's optimal level of initial software reliability investment?

Ultimately, we hope to provide answers to this question: Does the BBP serve as an effective and economically efficient way to improve software reliability, so that end users are better off by using a more reliable product and the total social welfare is improved as well?

We develop an analytical model that includes a platform, a third-party vendor, and multiple external hackers. We structure the BBP-associated costs and benefits for both the platform and the vendor, and model external hackers' behavior when BBP is provided. We study optimal decisions and interactions among these parties and obtain the following major results. First, we identify two key factors in the platform's (the vendor's) BBP launching (adoption) decision-making: the potential loss if a security breach happens and the vendor's initial reliability investment efficiency. For example, the vendor should only participate in the BBP if its expected potential loss is high and reliability investment efficiency is low. Second, we find that the platform's and the vendor's decisions might be inconsistent. For example, under certain conditions, the platform may prefer to have the BBP even if the potential loss is not high. The vendor, however, would not participate the BBP in such scenarios because the vendor's report-processing and bug-fixing costs cannot be fully offset by the BBP benefits. Therefore, the platform is over-incentivized compared to the vendor. We derived the conditions for this to happen. In addition, the conditions for the opposite – namely, the vendor is over-incentivized in BBP use, compared to the platform – are also presented in the paper. Third, the BBP, if adopted by both the vendor and platform, shall always bring higher utilities to them. It, however, might not always benefit end users of the software. We show different cases: (1) The BBP leads to an overall lower software reliability (and thus hurt end users) and consequently a lower total social welfare; (2) The BBP improves total social welfare –both the platform and vendor are better off, but it comes at the cost of decreased utility of end users (i.e., an overall reduced software reliability); And (3) the BBP is not only socially optimal, but also a Pareto improvement for all parties. For each case, we derive the conditions for it to happen and provide in-depth analysis of the underlying rationale.

The remainder of this paper is organized as follows. Section 2 provides a review of the related literature. Section 3 presents our model that captures the costs and benefits associated with BBPs for a software vendor and a platform, respectively. Section 4 analyzes the optimal BBP launching/participation strategies for the vendor and platform and examines the impact of BBP on the overall software reliability and social welfare. Finally, we conclude the paper by discussing its theoretical contributions and practical implications, and offer some future research directions in Section 5.

Literature Review

Our work relates to three streams of research. The first one is the BBP literature. Recent studies on BBPs mostly focus on its performance (e.g., Subramanian & Malladi, 2020; Walshe & Simpson, 2020; Aaltonen & Gao, 2021; Zhou & Hui, 2021; Zhou & Hui, 2022). For example, Zhou and Hui (2021) show that, under certain conditions, BBPs are economically beneficial to launch and can enhance the overall security. In addition, some research focuses on the BBP's optimal management policies. In this respect, Zhao et al. (2017) develop and investigate strategies that assign hackers to various BBPs and incentivize hackers to evaluate bug reports before submitting them. Different from all these works in BBP, ours is the first one to examine the optimal BBP adoption decisions for both the platform and the third-party software vendor.

In addition, other earlier works in this stream have mainly focused on examining the effects of in-house testing on software reliability (e.g., Ji et al., 2005; Jiang et al., 2012). More recently, researchers have started to look at external public testing methods (Jiang and Sarkar (2009)), which are closely related to this study. For instance, August and Marius (2013) analyze the impact of user error reporting on software reliability and investigate the optimal timing of software releases. Jiang et al.,(2017) focus on how beta testing improve software reliability. They develop models that help determine the optimal number of public beta testers and the ideal duration of testing, taking into account both reliability-related and market-related benefits, such as accelerating software diffusion and boosting user valuation of a product. Mehra and Saha (2018) discover that implementing public beta testing does not always result in a higher-quality product. Unlike other studies, we investigate the contributions of external professional hackers, particularly those who transit from malicious to ethical hacking, to the enhancement of software reliability, together with third-party vendor's software reliability investments.

The literature that studies on how digital platforms manage third-party software in order to maximize profit is also relevant to us. Previous research has primarily focused on whether platforms should be corporate with third parties (Boudreau, 2010; Parker et al., 2017; Karhu et al., 2018), as well as how to provide integration tools such as APIs and SDKs to third-party vendors (Tan et al., 2020; Xue et al., 2019). Furthermore, Huang et al. (2022) compare three regulatory strategies for digital platforms to manage third-party qualities: prohibiting access to low-quality complementors, subsidizing high-quality complementors, and developing its own third-party applications. Our paper differs from the existing literature in that we investigate when the platform shall adopt the BBP to prevent security incidents caused by third-party software.

Model Setup

Consider a digital platform (PF) on which the third-party vendor (VD) develops, lists, and sells its own software applications. The platform (PF) is considering launching a Bug Bounty Program (BBP) to improve the third-party software reliability. Under the BBP, individual external hackers are encouraged to report bugs that they find in the third-party software applications. After the vendor (VD) pays efforts to screen the reports, identify valid ones, and fix the bugs accordingly, the platform (PF) will give the external hacker monetary rewards. Leveraging the hacker community to enhance security, the BBP has the potential of benefiting the platform and the vendor. It, however, imposes extra costs on both, and might have an impact on the vendor's incentive of initial reliability investment in the software development stage. As a result, whether to offer a BBP (for the platform) and whether to participate in the BBP (for the vendor) are complicated strategic decisions. Throughout this paper, we denote L and NL as the platform's strategy of launching and not-launching BBP, and P and NP as the vendor's strategy of participating and not-participating BBP (in the case that BBP is launched by the platform). We then have two types of outcomes:

BBP when the strategies adopted by the platform and vendor are (L, P), and NBBP when the strategy combinations are (L, NP), (NL, P), or (NL, NP). We use BBP and NBBP as lower subscripts to indicate the two different outcomes.

External Hackers: We normalize the total number of initial external malicious hackers to be 1. Malicious hackers look for vulnerabilities and bugs in the third-party software applications; and once discovered, they launch attack, cause security breaches, and hurt both the platform and vendor. When there is BBP, some malicious hackers will convert to be ethical hackers. After discovering bugs in the software, ethical hackers will report to the platform for monetary rewards, instead of launching an attack. The platform will pay the reward r to the ethical hacker for each valid bug (i.e., after being verified by the vendor). It is intuitive that when this reward amount r is higher, more will turn to be ethical hackers. Previous empirical study (Zhao et al., 2015) demonstrates a significant positive linear correlation between the expected reward and the number of ethical hackers involved. Following it, we assume the number of existing hackers who convert from malicious to ethical as $\alpha_1 r$, where α_1 is malicious hackers' reward sensitivity coefficient, $0 < \alpha_1 r < 1$. Furthermore, the bug bounty program could potentially expand the pool of ethical hackers (i.e., attract new ethical hackers). Namely, the financial reward encourages participation from advanced users and security research institutions who otherwise would not. We denote the number of newly-attracted non-malicious hackers as $\alpha_2 r$, where α_2 is the reward sensitivity coefficient of these hackers. Denote n_e and n_m as the number of ethical and malicious hackers respectively. In the NBBP outcome, $n_e = 0$ and $n_m = 1$; and in the BBP outcome, $n_e = (\alpha_1 + \alpha_2)r$, $n_m = 1 - \alpha_1 r$.

The Vendor (VD): The reliability level of third-party software is denoted as S. Let 1 be the highest possible reliability level, $S < 1$. This assumption reflects the fact that there is no completely bug-free software in practice. Further, we define the vulnerability of the third-party software as $p = 1 - S$. Here, p also could be interpreted as the probability that external hackers discover valid bugs in the software.

The reliability of software, S, consists of three components. First, the software has an intrinsic reliability level S_0. It depends on the quality of the software itself, lies in the interior design of the application, and varies in terms of software types, programming languages and skills. For example, an online payment software product developed by a highly skilled developer team using a memory safe programming language usually has a high intrinsic reliability. In contrast, a reading software application designed by an individual developer and a loose programming language may naturally have a relatively low intrinsic reliability.

Second, to further improve reliability, the software vendor usually will make additional investments (such as ongoing software testing), which is referred as *vendor's reliability investment* in this study. We denote the vendor's reliability investment efforts as βz^2, where z is the resulted reliability improvement level, β is the investment cost coefficient that indicates the vendor's efficiency level in reliability investment, and the quadratic form represents the diminishing investment return. In the case of NBBP - either the platform does not launch a BBP or the vendor chooses not to participate, the software reliability level is the sum of these two components: $S_{NBBP} = S_0 + z_{NBBP}$, where the lower subscript NBBP indicates it is under the case that BBP is not in use.

In the case of BBP - when the platform launches BBPs and the vendor decides to participate, ethical hackers will discover and report bugs to the platform to gain rewards. After the vendor verifies valid bugs and fixes them, the software reliability level could be further improved. We recognize that it might take a while for the bugs to be identified and fixed, but this period should be transient. Therefore, we use upper superscripts B and A to indicate the software reliability level before and after such a transient period. As a result, the reliability level before the transient period (i.e., before bugs are reported and fixed) can be written as $S_{BBP}^B = S_0 + z_{BBP}$, where the lower subscript BBP indicates it is under the case that BBP is in use. During the transient period, there are $p_{BBP}^B n_e$ valid bugs to be fixed by the vendor, where $p_{BBP}^B = 1 - S_{BBP}^B$ is the probability that a hacker identifies a valid bug and $n_e = (\alpha_1 + \alpha_2)r$ is the total number of ethical hackers under the BBP reward amount r. Hence, after the transient period (i.e., after bugs have been fixed), the software reliability is improved and reaches to $S_{BBP}^A = S_{BBP}^B + p_{BBP}^B n_e f$, where f is the reliability level improvement due to the fix of each valid bug. We can also view f as the efficiency of BBP since it represents the unit reliability level improvement due to the BBP use. Consequently, the vulnerability of the third-party software reduces to $p_{BBP}^A = 1 - S_{BBP}^A$.

In reality, enhanced software reliability results in increased user satisfaction, which in turn attracts more customers to purchase the product and expands the user base (McDermott, 2023). In line with practice and previous literature (e.g., Garcia and Barry, 2007), we denote that the expected revenue generated from the third-party software as SR, where R is the highest possible revenue when the software is completely free of bugs (i.e., when $S = 1$). The expected revenue is shared by both the vendor and the platform: the vendor obtains θ percent and the platform $1 - \theta$ percent of it. In addition, if a malicious hacker discovers a valid bug, launches the attack, and results in security breaches, the software vendor and platform incur utility losses λ_{VD} and λ_{PF} respectively.

When BBP is offered by the platform, the vendor has two strategies to consider:

Not participate (NP): If the vendor chooses not to participate, then the outcome is NBBP. The vendor derives expected revenues $\theta S_{NBBP} R$ from the third-party software, pays the reliability investment cost βz_{NBBP}^2, and incurs the expected loss $p_{NBBP} n_m \lambda_{VD}$ caused by software breaches. The total payoff of the vendor is given by

$$\Pi_{NP}^{VD} = \theta S_{NBBP} R - (\beta z_{NBBP}^2 + p_{NBBP} n_m \lambda_{VD}). \tag{1}$$

Participate (P): If the vendor chooses to participate, then the outcome is BBP. The vendor derives expected revenues $\theta S_{BBP}^A R$ from the third-party software,[1] pays the reliability investment cost βz_{BBP}^2, and incurs potential loss $p_{BBP}^A n_m \lambda_{VD}$ caused by software breaches. Let c_p and c_f represent the vendor's unit processing and fixing cost, respectively. Thus, the vendor bears the total processing costs $n_e c_p$ for all bug reports, as well as the total fixing costs $p_{BBP}^B n_e c_f$ for all valid bug reports. The total payoff of the software vendor is given by

$$\Pi_{P}^{VD} = \theta S_{BBP}^A R - (\beta z_{BBP}^2 + p_{BBP}^A n_m \lambda_{VD} + n_e c_p + p_{BBP}^B n_e c_f). \tag{2}$$

The Platform (PF): Similarly, the platform can choose one of two options:

Not Launch (NL): If the platform does not launch the BBP, then the outcome is NBBP. In this case, the platform derives expected revenues $(1 - \theta) S_{NBBP} R$ from the third-party software and incurs the expected loss $p_{NBBP} n_m \lambda_{PF}$ caused by potential software breaches. The total payoff of the platform is

$$\Pi_{NL}^{PF} = (1 - \theta) S_{NBBP} R - p_{NBBP} n_m \lambda_{PF}. \tag{3}$$

Launch (L): If the platform launches the BBP, the outcome then depends on the vendor's choice. If the vendor chooses NP, the platform's payoff is the same as the above NL case, given in Equation (3). If the vendor chooses P, the platform derives expected revenues $(1 - \theta) S_{BBP}^A R$ from the third-party software, incurs the expected loss $p_{BBP}^A n_m \lambda_{PF}$, and pays hackers who reported valid bugs the BBP rewards $p_{BBP}^B n_e r$. As a result, the total payoff of the platform can be written as

$$\Pi_{L}^{PF} = (1 - \theta) S_{BBP}^A R - (p_{BBP}^A n_m \lambda_{PF} + p_{BBP}^B n_e r). \tag{4}$$

Appendix A provides a summary of our model notations and explanations.

Analysis and Results

Consider a digital platform (PF) on which the third-party vendor (VD) develops, lists, and sells its own software applications. The platform (PF) is considering launching a Bug Bounty Program (BBP) to improve the third-party. In this section, we first examine the optimal strategies of the software vendor in Section 4.1 and of the platform in Section 4.2. We then derive the conditions for BBP equilibrium and compare the software reliability level as well as social welfare under different equilibrium outcomes in Section 4.3.

Optimal Participation Strategy of the Vendor

After the software vendor decides to whether to participate the BBP program, it should determine the optimal investment level z_i in software reliability accordingly. The vendor maximizes its payoff:

$$\max_{z_i} \Pi_j^{VD}(z_i) \tag{5}$$

[1] We only consider the expected revenue after the transient period, in which the software reliability level has been improved to S_{BBP}^A.

$$s.t. z_i \geq 0$$

where $i = BBP\ or\ NBBP, j = P\ or\ NP$.

Solving Problem (5), we obtain the optimal reliability investment level of the vendor and present it in Proposition 1.

Proposition 1. (Optimal Reliability Investment Level)

a. *If the vendor participates in the BBP launched by the platform, the optimal reliability investment level is* $z^*_{BBP} = \frac{c_f r(\alpha_1+\alpha_2)+(1-fr(\alpha_1+\alpha_2))(R\theta+\lambda_{VD}(1-r\alpha_1))}{2\beta}$*; if it does not participate, the optimal reliability investment level is* $z^*_{NBBP} = \frac{R\theta+\lambda_V}{2\beta}$.

b. *If the BBP-related bug-fixing cost is low,* $c_f < \lambda_{VD} + f(R\theta + \lambda_{VD}(1 - r\alpha_1))$, $z^*_{BBP} < z^*_{NBBP}$*; otherwise,* $z^*_{BBP} \geq z^*_{NBBP}$.

Proposition 1 shows that when the third-party vendor chooses to participate in the BBP, its initial software reliability investment increases with the bug-fixing cost. It is because both the BBP and the vendor's initial investment serve to enhance the software reliability and thus help reduce the potential breach loss. If the BBP is "expensive," i.e., incurring in relatively high bug-fixing costs on the vendor, the vendor will instead increase its initial investment to improve the software's reliability level more. When the bug-fixing cost continues to increase and reach a certain threshold c_f, the vendor's optimal reliability investment level under participation becomes even higher than that under non-participation.

We also examine how malicious hackers' reward sensitivity level would affect the vendor's optimal reliability investment, as summarized in Corollary 1.

Corollary 1. (Impact of α_1 on Optimal Reliability Investment Level)

If malicious hackers are sensitive to BBP rewards, $\alpha_1 > \alpha_1^* = \frac{\lambda_{VD}+f(R\theta+\lambda_{VD}(1-\alpha_2))-c_f}{2fr\lambda_{VD}}$*, the vendor's optimal reliability investment level* z^*_{BBP} *increases with* α_1*; otherwise,* z^*_{BBP} *decreases with* α_1.

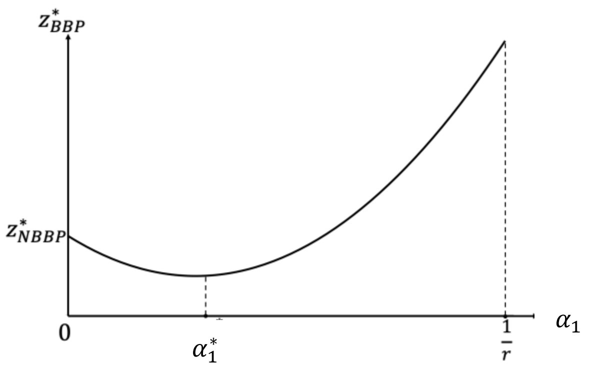

Figure 1. Impact of α_1 on Optimal Reliability Investment Level

We find that when the vendor participates in the BBP, the impact of malicious hackers' reward sensitivity α_1 on the optimal reliability investment level is non-monotonic, as illustrated in Figure 1. There is a threshold value α_1^*. When the malicious hackers' reward sensitivity α_1 is relatively low, $\alpha_1 < \alpha_1^*$, the vendor should decrease the reliability investment level as more hackers convert to be ethical. In this case, malicious hackers are not responsive to BBP rewards actively and thus are difficult to be converted. As a result, the marginal benefits of having more ethical hackers are high, which could benefit the vendor more significantly than the initial reliability investment. However, if the malicious hackers' reward sensitivity exceeds the certain threshold α_1^*, the number of participating ethical hackers has grown excessively large and the marginal benefits of converting more ethical hackers become low. Therefore, if α_1^* increases further, the vendor should increase its software reliability level so that participating hackers will discover fewer valid bugs, allowing the vendor to save enough money on BBP-related costs (i.e., report-processing and bug-fixing costs).

Next, we substitute the optimal reliability investment (z_{BBP}^* and z_{NBBP}^*) into the vendor's profit function (Equations (1) and (2)), to obtain the vendor's total payoff under the BBP and NBBP outcomes. Comparing the vendor's payoffs, we obtain its BBP participation condition in Proposition 2.

Proposition 2. (Software Vendor's Participation Decision)

The third-party software vendor will participate in the BBP if and only if: (1)its potential loss due to security breaches is high, $\lambda_{VD} > \lambda_{VD1}$ *AND (2) its reliability investment efficiency is low,* $\beta > max\,\{\beta_{VD}, \beta_1\}$.

All threshold values in Propositions are presented in the Appendix B.

Proposition 2 reveals the fundamental economic rationale of the third-party software vendor's BBP participation decision. Intuitively, when the potential loss of security breach is significant, the vendor needs to actively search ways to improve the software reliability. For the vendor, there are two viable ways: increasing its initial reliability investment or leveraging hacker community through participating BBP. If the vendor's reliability investment efficiency is high, it will opt for the first way only (i.e., not participating in the BBP) because its reliability investment alone is sufficient to reduce potential loss, which also avoids the BBP-related costs. But if the vendor's reliability investment is inefficient, the BBP needs to adopt the BBP as a way to complement its own reliability investment.

Corollary 2 below presents an interesting finding.

Corollary 2. (Reliability Investment Level Comparison)

When the vendor decides to participate the BBP, it will always decrease the initial reliability investment.

Recall that Proposition 1 states that when the bug-fixing cost is high, the level of vendor's reliability investment can be higher in the BBP outcome than NBBP outcome. Proposition 2, however, suggests that under such a scenario, the vendor then should choose not to participate in the BBP because participation will also result in lower profit for it. By integrating Propositions 1 and 2, we derive Corollary 2. This indicates that whenever the vendor chooses to participate in the BBP (thus to gain higher total payoffs), it shall reduce its own initial investment in the software reliability because now it could leverage BBP to enhance the software later.

Optimal Launch Strategy of the Platform

BBP imposes additional costs on the digital platform. We analyze when it is profitable for the platform to launch the BBP for the third-party vendor. We have the following findings.

Proposition 3. (Platform's Launch Decision)

The platform will choose to launch the BBP:

a. *when the platform's potential loss is high,* $\lambda_{PF} \geq \lambda_{PF1}$, *and the vendor's reliability investment efficiency is low,* $\beta > \beta'$ *where* $\beta' = \begin{cases} \beta_1 & if\ \lambda_{VD} < \lambda_{VD1} \\ \beta_{PF} & if\ \lambda_{VD} \geq \lambda_{VD1} \end{cases}$.
b. *when the platform's potential loss is low,* $\lambda_{PF} < \lambda_{PF1}$, *and the vendor's potential loss is low* $\lambda_{VD} < \lambda_{VD1}$ *and reliability investment efficiency is moderate* $\beta \in [\beta_1, \beta_{PF}]$.

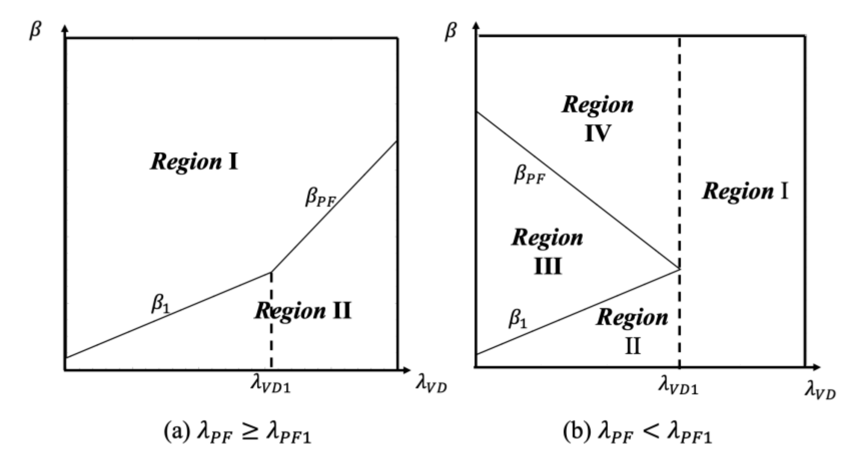

Figure 2. The Platform's Launch Strategies of the BBP

Proposition 3 describes two scenarios under which the platform should launch the BBP to gain higher payoffs. We illustrate them in Figure 2, and provide explanations as follows.

Intuitively, the platform is more willing to launch BBP if it expects a high potential breach loss and meanwhile when the third-party vendor is not efficient in making software reliability investment. This is the case described by Proposition 3a. Although the platform needs to pay monetary rewards to ethical hackers, the benefits from the threat reduction and decreased potential loss make it worthwhile. This is the region I in Figure 2(a).

The second case when the platform should opt for launching BBP is described by Proposition 3b and demonstrated in region III in Figure 2(b). Note that in this case, the platform expects a small loss even if a breach happens. Then what motivates the platform to launch BBP, given there are additional costs of doing so? This is a less intuitive but more interesting case. To understand the platform's choices in $\lambda_{PF} < \lambda_{PF1}$, we must take the vendor's reaction into considerations as well. Recall the software vendor's optimal reliability investment level, $z^*_{BBP} = \frac{c_f r(\alpha_1+\alpha_2)+(1-fr(\alpha_1+\alpha_2))(R\theta+\lambda_{VD}(1-r\alpha_1))}{2\beta}$ (Proposition 1). Note that z^*_{BBP} increases in λ_{VD}, the vendor's potential loss due to security breaches. Hence, when $\lambda_{VD} > \lambda_{VD1}$, the vendor's initial reliability investment is sufficiently high so that the platform will not have interest in launching BBP to further improving software reliability, which is the region I of Figure 2(b).

Next, we analyze the case of $\lambda_{VD} \leq \lambda_{VD1}$. Note that z^*_{BBP} decreases in β, vendor's reliability investment efficiency. λ_{VD} and β thus have opposite effects on the optimal reliability investment level z^*_{BBP}.[2] As a result, when both λ_{VD} and β are small, namely, $\lambda_{VD} < \lambda_{VD1}$ and $\beta < \beta_1$, vendor chooses to invest in software reliability at a moderate level. For the platform, the benefits of launching BBP are from reducing the potential loss caused by the security breaches. Since the platform's loss is low when a breach happens, it is likely that the platform would consider the vendor's initial reliability investment as sufficient. Thus, the platform will not have interest in launching BBP, as shown in the region II of Figure 2(b).

When β increases to a moderate level, $\beta \in [\beta_1, \beta_{PF}]$, the optimal reliability investment of the vendor decreases accordingly and becomes insufficient for the platform in the sense that the low software reliability induces too many software vulnerabilities and meanwhile generates too little revenue. Therefore, the platform now is willing to invest in the BBP to increase software reliability, expecting that it will obtain more revenues from end users and thus become more profitable. This is what happens in the region III in Figure 2(b).

Finally, if the vendor's reliability investment efficiency β becomes very low, $\beta > \beta_{PF}$, the third-party vendor will choose a very low reliability level in the first place. If the platform launches the BBP for the vendor

[2] Based on constraints illustrated in our model setup, $1 - fr(\alpha_1 + \alpha_2) > 0$ and $1 - r\alpha_1 \geq 0$.

under such a situation, it must pay significant rewards for many reported bugs because of the poor initial software reliability. The cost of launching a BBP exceeds the revenue gains and breach loss savings, the platform again loses interest in launching BBP, as shown in region IV in Figure 2(b).

Combining the findings from Propositions 2 and 3, we conclude the equilibrium outcome conditions in Proposition 4.

Proposition 4. (Equilibrium Outcome)

When the potential loss of both the vendor and platform are high, $\lambda_{VD} \geq \lambda_{VD1}$ *and* $\lambda_{PF} \geq \lambda_{PF1}$*, and when the vendor's reliability investment efficiency is low,* $\beta > \beta_{BBP} = max\{\beta_{VD}, \beta_{PF}\}$*, the equilibrium outcome is BBP; Otherwise, it is NBBP.*

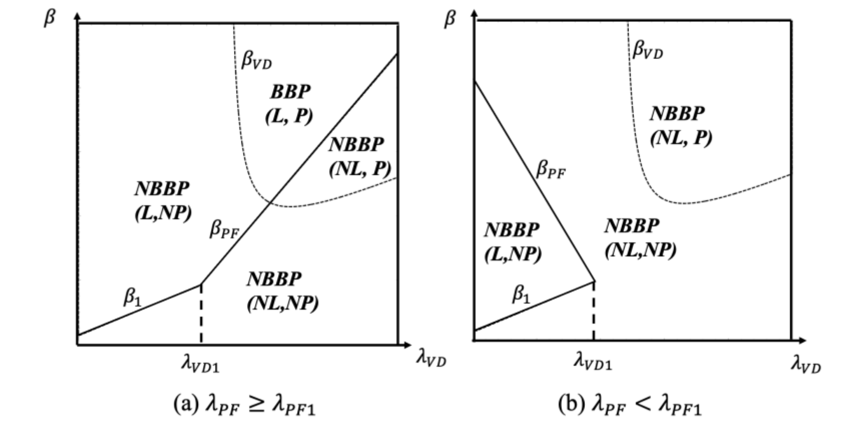

Figure 3. Conditions for BBP and NBBP Outcome

Figure 3 depicts the regions of two equilibrium outcomes (BBP and NBBP), as well as various strategy combinations of the platform and vendor that lead to the respective equilibrium outcome –the strategy pair of (L, P) that results in BBP, and (L, NP), (NL, P), and (NL, NP) that result in NBBP.

The BBP equilibrium appears only in the upper-right region of Figure 3(a), which is when security breach will cause significant loss for both the vendor and platform and when the vendor cannot efficiently invest in software reliability alone. As a result, both the platform and vendor are willing to pay extra efforts to leverage external hackers' capability to improve software security. This finding validates Google's policy of launching the BBP primarily for third-party software with more than 100 million installs (Vaas, 2019), which could potentially result in huge financial loss if a breach happens on such large-installation-base applications.

In regions of NBBP as the equilibrium, some are because the platform's and the vendor's incentives are not well aligned. For example, in the region (L, NP) in Figure 3(a), the platform has an incentive to launch a BBP given the vendor's low reliability investment efficiency, but the vendor chooses not to participate the BBP because its potential loss due to security breach is estimated to be low. We have seen such cases in practice. For example, SlimSocial (Facebook, 2023), a third-party Facebook software that helps users to launch Facebook quickly, decided not to participate the BBP offered by Facebook, because the expected loss on the vendor's side, even if an attack happens and succeeds, won't be significant.

Similarly, we also observe that sometimes the vendor has a stronger incentive than the platform to use BBP, such as the (NL, P) region in Figure 3(b), in which the vendor is facing large breach loss and low reliability investment efficiency and thus would like to adopt BBP, while the platform is not willing to launch BBP due to its expected low breach loss.

Software Reliability Level and Social Welfare Analysis

Although BBP relies on external hackers to improve the software reliability and reduce the chance of software breach loss, it also has an impact on the vendor's behavior. We have shown that the vendor will always reduce its initial software reliability investment once the BBP is in use. As a result, it is unclear whether the overall software reliability in the BBP outcome gets improved, compared to the NBBP outcome. Consequently, it is also unclear whether end users are enjoying a more secure environment, and whether the total social welfare is enhanced. In this subsection, we examine and compare the software reliability level and social welfare under BBP and NBBP outcomes.

We present the software reliability comparison finding in Proposition 5.

Proposition 5. (Reliability Level Comparison)

Only when the vendor's reliability investment efficiency is low, $\beta > \beta_{RL}$, the use of BBP helps to increase the overall software reliability; otherwise, it decreases the overall software reliability.

This is an important finding. It shows that the use of BBP, even though it decreases the number of malicious hackers and imposes additional costs for bug-searching-and-fixing, might reduce the third-party software reliability and result in a more vulnerable marketplace. It is because after adopting BBP, the vendor will always reduce its own investment in software reliability. Investing less in the initial software design stage, the vendor expects to leverage more on ethical hackers to discover software vulnerabilities in the later stage. This, however, only works if the vendor's reliability investment efficiency is low, so that BBP is more economical and effective in improving the software reliability level. Otherwise, the reduction in vendor's initial reliability investment cannot be fully offset by the reliability improvement due to the BBP use, which hence results in a decreased in overall software reliability.

Proposition 5 reports the comparison of reliability levels without taking into account the platform or vendor's decisions. In Corollary 3, we further specify the parameter conditions under which BBP is adopted by both the platform and vendor but results in a lower overall software reliability.

Corollary 3. (Conditions when BBP Reduces the Overall Reliability Level)

When the bug processing cost under BBP is low, $c_p < c_{prl}$, and the platform's potential loss due to breaches is high, $\lambda_{PF} > \lambda_{PFrl}$, in the region of $\beta_{BBP} < \beta < \beta_{RL}$, both the platform and vendor will choose to use BBP, which however results in a lower overall software reliability level than the NBBP outcome.

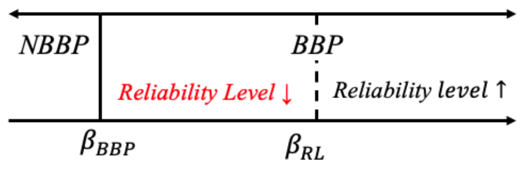

Figure 4. Software Reliability Level

A low processing cost of bug reports motivates the software vendor to participate in the BBP and reap the benefits of threat reduction at a low cost. Meanwhile, a large potential loss caused by security breaches makes the BBP attractive to the platform. In such a situation, the platform and vendor are more likely to launch and participate in the BBP.[3] We can prove that there exists a non-empty region of $(\beta_{BBP}, \beta_{RL})$, as shown in Figure 4, such that the BBP is the equilibrium outcome and it results in lower overall software reliability than NBBP.

[3] The threshold value β_{BBP} becomes smaller and thus leads to a higher chance of BBP equilibrium outcome.

Next, we study the social welfare change after using BBP. Total social welfare is the sum of software end users' surplus, the third-party vendor's payoff and the platform's payoff. Let vS_i denote the value derived by software end users gain from consuming the software product, where $i = BBP \text{ or } NBBP$. The total social welfare can be written as

$$W_{NBBP} = \Pi_{NP}^{VD} + \Pi_{NL}^{PF} + (v - R)S_{NBBP} \tag{6}$$

and

$$W_{BBP} = \Pi_{P}^{VD} + \Pi_{L}^{PF} + (v - R)S_{BBP}^{A} \tag{7}$$

in the NBBP and BBP outcome respectively.

Proposition 6 reports our comparison result.

Proposition 6. (Social Welfare Comparison)

Only when the vendor's reliability investment efficiency is low, $\beta > \beta_{SW}$, the use of BBP increases total social welfare; otherwise, it decreases the total social welfare.

When the vendor's reliability investment is inefficient, BBP becomes economically attractive. We find that as β is sufficiently large, $\beta > \beta_{SW}$, it is socially optimal to use BBP. BBP will lead to higher end user surplus, and higher payoffs for the vendor and platform compared to the NBBP outcome.

Proposition 6 also suggests that if β does not exceed the threshold value β_{SW}, BBP will decrease the total social welfare. Corollary 4 below highlights the scenarios under which that the platform and the vendor will choose to launch and participate BBP but the use of BBP indeed is not socially optimal.

Corollary 4. (Conditions when BBP Reduces the Social Welfare)

When the bug processing cost under BBP is low, $c_p < c_{psw}$, the platform's potential loss due to breaches is high, $\lambda_{PF} > \lambda_{PFsw}$, and end user's value of consuming the software is high, $v > v_{sw}$, in the region of $\beta_{BBP} < \beta < \beta_{SW}$, both the platform and vendor will choose to use BBP, which however results in a lower total social welfare than the NBBP outcome.

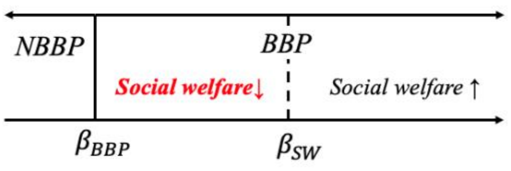

Figure 5. Social Welfare

The intuition here is similar to that for Corollary 3. As explained, a lower bug processing cost and a higher potential loss due to security breaches result in a lower threshold value β_{BBP}, which means that the BBP is more likely to be the equilibrium outcome while also more likely to result in lower overall software reliability. Low software reliability hurts end users. In particular, end users' surplus (vS_{BBP}^{A}) will get a large reduction if the parameter v value is big.[4] In such a case, we can prove there exists an non-empty region of $(\beta_{BBP}, \beta_{SW})$, as shown in Figure 5, such that BBP is the equilibrium outcome in this region and it results in lower social welfare than NBBP. Note that in this region, both the platform and vendor are better off by using BBP, but end users are worse off.

[4] As c_p decreases while λ_{PF} and v increase, the threshold value β_{BBP} becomes smaller and β_{SW} becomes larger, and eventually $\beta_{BBP} < \beta_{SW}$ holds.

Finally, we combine all above findings to offer a comprehensive view of how the BBP use would affect software reliability and social welfare.

Proposition 7. (Reliability Level and Social Welfare)

When the bug processing cost is low, $c_p < min\,\{c_{p_{sl}}, c_{p_{sw}}\}$, the platform's potential loss due to security breaches is high, $\lambda_{PF} > max\,\{\lambda_{PF\_sw}, \lambda_{PF\_sw}\}$, and users' value of consuming the software is high, $v > v_{sw}$, the BBP

a. reduces both software reliability and social welfare if $\beta < \beta_{SW}$;

b. reduces software reliability level but increases social welfare if $\beta_{SW} < \beta < \beta_{RL}$;

c. increases both software reliability and social welfare if $\beta > \beta_{RL}$.

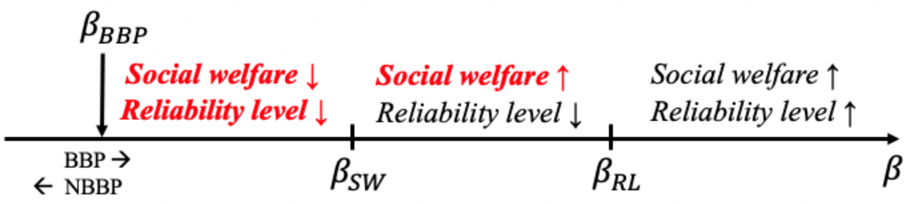

Figure 6. The Impact of BBP on Social Welfare and Reliability Level

We use Figure 6 to illustrate Proposition 7. When the third-party vendor is very efficient in reliability investment, $\beta < \beta_{BBP}$, either the platform or the vendor, or both will find that the benefits brought on by BBP cannot offset the extra costs imposed on them. Hence, the outcome is NBBP. When the vendor's reliability investment efficiency decreases, $\beta \in [\beta_{BBP}, \beta_{SW}]$, both the vendor and platform will find it profitable to use BBP. But in this region, because the vendor decreases initial investment, end users are worse off and total social welfare turns out to be lower. This is the worst-case scenario that the suppliers (both the vendor and platform) are over-incentivized. The adoption of a reliability-improving strategy (BBP) in fact results in a less secure environment. This finding implies that software vendors with a strong and skilled security team may not need BBP. The use of BBP will make these vendors under-invest in their software reliability at the first place, eventually hurting end users' utility and leading to a lower social welfare.

If the vendor's reliability investment efficiency keeps reducing, in the region of $\beta \in [\beta_{SW}, \beta_{RL}]$, users are still worse off for using a lower reliability product but the total social welfare is higher compared to the NBBP outcome; and when β increases further, beyond β_{RL}, BBP leads to both higher software reliability and higher social welfare. This is because the software vendor's initial reliability investment is not sufficient. BBP now serves as an effective and economically efficient way to improve the overall reliability of the software. End users now are better off as well.

Conclusion

Consider a digital platform (PF) on which the third-party vendor (VD) develops, lists, and sells its own software applications. In this paper, we model the interactions between a platform and a third-party software vendor in considering using BBP to improve software reliability. Expecting that the BBP could reduce security threat, increase software revenue, but also incur additional costs, the platform must make the decision of whether and when to launch the BBP. Similarly, the vendor must consider whether to participate in the BBP and if so, how should it adjust the initial reliability investment level.

Our findings show that not all vendors prefer to participate the BBP, and when they do, they will reduce their initial reliability investment level as a response. Meanwhile, we find that that even when the potential loss of the vendor and the platform are relatively low, the platform may still have an incentive to launch a BBP for those vendors who have moderate reliability investment efficiency. However, vendors could refuse to participate in such a BBP. We conclude that the BBP is profitable for both the vendor and platform only when the vendor is inefficient in the software reliability investment and when the potential loss for both are high.

While a common belief is that BBP could leverage external hackers' efforts to deliver high software reliability, this paper provides some counter-intuitive but reasonable findings. We show that software reliability might be lower with the BBP than the without, especially when the vendor's own reliability investment efficiency is sufficiently high. In such a case, although both the platform and the vendor greatly benefit from the threat reduction due to BBP use, software end users are worse off because BBP reduces the vendor's software reliability investment incentive in the first place. As a result, users are using a less reliable product and their surpluses decrease, which consequently could even lead to a lower social welfare. This finding suggests that the regulator should pay extra attention to such BBPs that are used by software vendors with relatively professional security teams.

Our study can be extended in several ways. First, we investigate the optimal BBP adoption strategies for one platform and one vendor. A possible future direction is to employ the optimal BBP launch strategies for one platform and multiple vendors. Further, in this study, we assume bug bounty reward amounts are exogenously given. An extension to our work is to incorporate bug bounty reward as the platform's endogenous decision.

Acknowledgements

The authors wish to acknowledge the National Natural Science Foundation of China (72231004, 71871155), China Scholarship Council (202206250105), and the Shandong Industrial Internet Innovation and Entrepreneurship Community for financial support. Nan Feng is the corresponding author.

References

Aaltonen, A., & Gao, Y. (2021). Does the outsider help? The impact of bug bounty programs on data breaches (SSRN Working Paper No. 3908761). *Temple University*. **https://ssrn.com/abstract=3908761.**

Apple. (2023). Apple security research device program. *Apple*. **https://security.apple.com/research-device/.**

August, T., & Niculescu, M. F. (2013). The influence of software process maturity and customer error reporting on software release and pricing. *Management Science, 59*(12), 2702-2726. **https://doi.org/10.1287/mnsc.2013.1728**

Boudreau, K. (2010). Open platform strategies and innovation: Granting access vs. devolving control. *Management Science, 56*(10):1849–1872. **https://pubsonline.informs.org/doi/abs/10.1287/mnsc.1100.1215**

Caminade, J., & Wartburg, M. (2022). The success of third-party apps on the app store. Analysis Group. **https://www.apple.com/newsroom/pdfs/the-success-of-third-party-apps-on-the-app-store.pdf.**

Facebook. (2022). Meta bug bounty program info. *Facebook*. **https://m.facebook.com/whitehat/info.**

Garcia, A. and Horowitz, B. (2007). The potential for underinvestment in internet security: implications for regulatory policy. *Journal of Regulatory Economics, 31*, 37-55. **https://doi.org/10.1007/s11149-006-9011-y**

Google. (2023). Google play security reward program rules. *Google*. **https://bughunters.google.com/about/rules/5604090422493184.**

Huang, P., Lyu, G., & Xu, Y. (2022). Quality regulation on two-sided platforms: Exclusion, subsidization, and first-party applications. *Management Science, 68*(6):4415–4434. **https://doi.org/10.1287/mnsc.2021.4075**

Ji, Y., Mookerjee, V. S, & Sethi, S. P. (2005). Optimal software development: A control theoretic approach. *Information Systems Research, 16*(3), 292-306. **https://doi.org/10.1287/isre.1050.0059**

Jiang, Z., & Sarkar, S. (2009). Speed matters: The role of free software offer in software diffusion. *Journal of Management Information Systems, 26*(3), 207-240. **https://doi.org/10.2753/MIS0742-1222260307**

Jiang, Z., Sarkar, S., & Jacob, V. S. (2012). Postrelease testing and software release policy for enterprise-level systems. *Information Systems Research, 23*(3-part-1), 635-657. **https://pubsonline.informs.org/doi/abs/10.1287/isre.1110.0379**

Jiang, Z., Scheibe, K. P., Nilakanta, S., & Qu, X. (2017). The economics of public beta testing. *Decision Sciences, 48*(1), 150-175. **https://doi.org/10.1111/deci.12221**

Karhu, K., Gustafsson, R., & Lyytinen, K. (2018). Exploiting and defending open digital platforms with boundary resources: Android's five platform forks. *Information Systems Research, 29*(2):479–497. **https://pubsonline.informs.org/doi/10.1287/isre.2018.0786**

Kumar, M. (2019). Facebook now pays hackers for reporting security bugs in 3rd-party apps. *The Hacker News*. **https://thehackernews.com/2019/10/facebook-apps-bug-bounty.html.**

Lapowsky, I. (2019). In latest Facebook data exposure, history repeats itself. *Wired*. **https://www.wired.com/story/facebook-apps-540-million-records/.**

McDermott, C. (2023). How quality became a growth driver. *Global App Testing*. **https://www.globalapptesting.com/blog/how-quality-became-a-growth-driver**

Mehra, A., & Saha, R. L. (2018). Utilizing public betas and free trials to launch a software product. *Production and Operations Management, 27*(11), 2025-2037. **https://onlinelibrary.wiley.com/doi/10.1111/poms.12740.**

O'driscoll, A. (2023). 30+ data breach statistics and facts. *Comparitech*. **https://www.comparitech.com/blog/vpn-privacy/data-breach-statistics-facts/.**

Parker, G., Van, A. M., & Jiang, X. (2017). Platform ecosystems: How developers invert the firm. *MIS Quarterly, 41*(1):255–266. **https://dl.acm.org/doi/10.5555/3177663.3177677**

SlimSocial. (2023). SlimSocial Project. *Facebook*. **https://www.facebook.com/SlimSocialProject/.**

Subramanian, H.C., & Malladi, S. (2020). Bug bounty marketplaces and enabling responsible vulnerability disclosure: An empirical analysis. *Journal of Database Management, 31*(1), 38-63. **http://dx.doi.org/10.4018/JDM.2020010103**

Tan, B., Anderson, E.G., & Parker, G.G. (2020). Platform pricing and investment to drive third-party value creation in two-sided networks. *Information Systems Research, 31*(1):217–239. **https://doi.org/10.1287/isre.2019.0882**

TikTok. (2023). TikTok bug bounty program policy. *HackerOne*. **https://hackerone.com/tiktok?type=team.**

Vaas, L. (2019). Google throws bug bounty bucks at mega-popular third-party apps. *Naked Security*. **https://nakedsecurity.sophos.com/2019/09/02/google-throws-bug-bounty-bucks-at-mega-popular-third-party-apps/.**

Walshe, T., & Simpson, A. (2020). An empirical study of bug bounty programs. *2020 IEEE 2nd International Workshop on Intelligent Bug Fixing (IBF)*. **https://doi.org/10.1109/IBF50092.2020.9034828**

Xue, L., Song, P., Rai, A., Zhang, C., Zhao, X. (2019). Implications of application programming interfaces for third-party new app development and copycatting. *Production and Operations Management, 28*(8):1887–1902. **https://doi.org/10.1111/poms.13021**

Zhao, M., Aron L., & Jens G. (2017). Devising effective policies for bug-bounty platforms and security vulnerability discovery. *Journal of Information Policy, 7*(1), 372-418. **https://doi.org/10.5325/jinfopoli.7.2017.0372**

Zhao, M., Grossklags, J., & Liu, P. (2015). An empirical study of web vulnerability discovery ecosystems. *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. **https://doi.org/10.1145/2810103.2813704**

Zhou, J., & Hui, K.L. (2021). Sleeping with the enemy: An economic and security analysis of bug bounty programs (HKUST Business School Research Paper No. 2021-038). *Hong Kong University of Science and Technology*. **https://ssrn.com/abstract=3940307.**

Zhou, J., & Hui, K. L. (2022). Strategic interaction between crowd and in-house contributions: Evidence from the internet bug bounty program (HKUST Business School Research Paper No. 2022-055). *Hong Kong University of Science and Technology*. **https://ssrn.com/abstract=4074182.**

Appendix

Appendix A. Model Notations and Definitions

Notations	**Definitions**
r	Bug bounty reward for each valid bug reported by ethical hackers
$\alpha_1(\alpha_2)$	Malicious (Non-malicious) hackers' reward sensitivity coefficient
$n_e(n_m)$	The number of ethical (malicious) hackers in BBP and NBBP scenario; In NBBP: $n_e = 0, n_m = 1$; In BBP: $n_e = (\alpha_1 + \alpha_2)r, n_m = 1 - \alpha_1 r$
S_0	Intrinsic reliability level of the third-party software
β	Vendor's reliability investment cost per unit reliability level squared
BBP, NBBP	Lower scripts to indicate the outcome (with or without BBP).
S_i	Reliability level of software in case $i = BBP$ or $NBBP, S_i < 1$. In the BBP outcome, $S_{BPP}^{B}(S_{BBP}^{A})$ denotes the reliability level of software before (after) the transient stage.
p_i	Vulnerability of software in case $i = BBP$ or $NBBP$; $p_i = 1 - S_i$.
f	Reliability level improvement due to fixing per valid bug. It measures the BBP's efficiency.
z_i	Reliability level improvement due to the vendor's reliability investment, under cases $i = BBP$ or $NBBP$. It measures the efficiency of vendor's reliability investment.
c_p	Vendor's (unit) cost of processing and verifying one bug report
c_f	Vendor's (unit) cost of fixing one valid bug
$\lambda_{PF}(\lambda_{VD})$	Potential loss of the platform (vendor) caused by software breaches
R	Maximum revenue earned from the third-party software (under the hypothetical situation of $S = 1$)
θ	Percentage of revenue obtained by the software vendor; $1 - \theta$ represents percentage of revenue obtained by the platform

Table 1. Model Notations and Definitions

Appendix B. Threshold Values

1. $\beta_1 = max\, \{\frac{R\theta+\lambda_{VD}}{2(1-S_0)}, \frac{r(\alpha_1+\alpha_2)c_f+(1-fr(\alpha_1+\alpha_2))(R\theta+\lambda_{VD}(1-r\alpha_1))}{2(1-S_0)}\}$

2. $\beta_{VD} = \frac{rc_f^2(\alpha_1+\alpha_2)^2+2c_f(\alpha_1+\alpha_2)(1-fr(\alpha_1+\alpha_2))(R\theta+(1-r\alpha_1)\lambda_{VD})-(R\theta(2-fr(\alpha_1+\alpha_2))+(2+fr^2\alpha_1(\alpha_1+\alpha_2)-r(\alpha_1+f\alpha_1+f\alpha_2))\lambda_{VD})(\alpha_1\lambda_{VD}+f(\alpha_1+\alpha_2)(R\theta+(1-r\alpha_1)\lambda_{VD}))}{4(1-S_0)(\alpha_1+\alpha_2)c_f+4(c_p(\alpha_1+\alpha_2)-(1-S_0)(\lambda_{VD}+f(\alpha_1+\alpha_2)(R\theta+\lambda_{VD}-r\alpha_1\lambda_{VD})))}$

3. $\beta_{PF} = \frac{1}{2(1-S_0)(f(\alpha_1+\alpha_2)(R(1-\theta)+\lambda_{PF})+\alpha_1\lambda_{PF}-r(\alpha_1+\alpha_2)(1+f\alpha_1\lambda_{PF}))}(R\theta(2f(\alpha_1+\alpha_2)(R(1-\theta)+\lambda_{PF})+\alpha_1\lambda_{PF}) - rR\theta(\alpha_1+\alpha_2)(1+f^2(\alpha_1+\alpha_2)(R(1-\theta)+\lambda_{PF})-2f\alpha_1\lambda_{PF})+c_f(\alpha_1+\alpha_2)(-r^2(\alpha_1+\alpha_2)+R(1-\theta)(fr(\alpha_1+\alpha_2)-1)-(1-r\alpha_1)(1-fr(\alpha_1+\alpha_2))\lambda_{PF})-fr^3\alpha_1(\alpha_1+\alpha_2)^2(1+f\alpha_1\lambda_{PF})\lambda_{VD}+(R(1-\theta)(\alpha_1+2f\alpha_1+2f\alpha_2)+2(\alpha_1+f\alpha_1+f\alpha_2)\lambda_{PF})\lambda_{VD}-r((\alpha_1+\alpha_2)(1-fR(1-\theta)((2+f)\alpha_1+f\alpha_2))-((1+f(4+f))\alpha_1^2+2f(2+f)\alpha_1\alpha_2+f^2\alpha_2^2)\lambda_{PF})\lambda_{VD}+r^2(\alpha_1+\alpha_2)(fR\theta(\alpha_1+\alpha_2)+\alpha_1\lambda_{VD}+f(\alpha_1+\alpha_2+2\alpha_1^2\lambda_{PF})\lambda_{VD}+f^2\alpha_1(\alpha_1+\alpha_2)(2\lambda_{PF}\lambda_{VD}+R(\theta\lambda_{PF}+\lambda_{VD}-\theta\lambda_{VD}))))$

4. $\beta_{RL} = \frac{c_f(\alpha_1+\alpha_2)(fr(\alpha_1+\alpha_2)-1)+\alpha_1\lambda_{\mathrm{VD}}+(2f(\alpha_1+\alpha_2)-f^2r(\alpha_1+\alpha_2)^2)(R\theta+\lambda_{\mathrm{VD}}-r\alpha_1\lambda_{\mathrm{VD}})}{2f(1-S_0)(\alpha_1+\alpha_2)}$

5. $\beta_{SW} = \frac{1}{4((1-S_0)(\alpha_1(\lambda_{PF}+\lambda_{VD})+f(\alpha_1+\alpha_2)(v+\lambda_{PF}+\lambda_{VD})-r(\alpha_1+\alpha_2)(1+f\alpha_1(\lambda_{PF}+\lambda_{VD})))-c_p(\alpha_1+\alpha_2)-c_f(1-S_0)(\alpha_1+\alpha_2))}[-rc_f^2(\alpha_1+\alpha_2)^2 - fr^3\alpha_1(\alpha_1+\alpha_2)^2\lambda_{VD}(2+f\alpha_1(2\lambda_{PF}+\lambda_{VD}))+2(R\theta\alpha_1\lambda_{PF}+\alpha_1\lambda_{VD}(v+2\lambda_{PF}+\lambda_{VD})+f(\alpha_1+\alpha_2)(R\theta+\lambda_{VD})(2v-R\theta+2\lambda_{PF}+\lambda_{VD}))-2c_f(\alpha_1+\alpha_2)(v-frv(\alpha_1+\alpha_2)+\lambda_{PF}+\lambda_{VD}-r(\alpha_1+f\alpha_1+f\alpha_2)(\lambda_{PF}+\lambda_{VD})+r^2(\alpha_1+\alpha_2)(1+f\alpha_1(\lambda_{PF}+\lambda_{VD})))+2r^2(\alpha_1+\alpha_2)(\alpha_1\lambda_{VD}+f^2\alpha_1(\alpha_1+\alpha_2)(R\theta\lambda_{PF}+\lambda_{VD}(v+2\lambda_{PF}+\lambda_{VD}))+f(R\theta(\alpha_1+\alpha_2)+\lambda_{VD}(\alpha_1+\alpha_2+\alpha_1^2(2\lambda_{PF}+\lambda_{VD}))))+r(f^2R^2\theta^2(\alpha_1+\alpha_2)^2-2R\theta(\alpha_1+\alpha_2)(1+2f\alpha_1\lambda_{PF}+f^2(\alpha_1+\alpha_2)(v+\lambda_{PF}))-\lambda_{VD}(\alpha_1^2(2f(2+f)v+2\lambda_{PF}+\lambda_{VD}+f(4+f)(2\lambda_{PF}+\lambda_{VD}))+\alpha_2(2+f^2\alpha_2(2(v+\lambda_{PF})+\lambda_{VD}))+2\alpha_1(1+f\alpha_2(2(1+f)v+(2+f)(2\lambda_{PF}+\lambda_{VD})))))]$

6. $\lambda_{VD1} = \frac{(c_f-fR\theta)(\alpha_1+\alpha_2)}{\alpha_1(1+f(1-r\alpha_1))+f(1-r\alpha_1)\alpha_2}$

7. $\lambda_{PF1} = \frac{(r-fR(1-\theta))(\alpha_1+\alpha_2)}{\alpha_1(1+f-fr\alpha_1)+f(1-r\alpha_1)\alpha_2}$

8. $\lambda_{PFrl} = \frac{r(\alpha_1+\alpha_2)}{\alpha_1(1-fr(\alpha_1+\alpha_2))}$

9. $\lambda_{PFsw} = \frac{1}{2(1-S_0)\alpha_1(1-fr(\alpha_1+\alpha_2))(c_f(\alpha_1+\alpha_2)-\alpha_1\lambda_{\mathrm{VD}}-f(\alpha_1+\alpha_2)(R\theta+\lambda_{\mathrm{VD}}-r\alpha_1\lambda_{\mathrm{VD}}))}(c_f^2(1-S_0)(\alpha_1+\alpha_2)^2(2-fr(\alpha_1+\alpha_2))-(1-S_0)((\alpha_1+\alpha_2)(2r+fR\theta(fr(\alpha_1+\alpha_2)-2))+(\alpha_1(1+f(1-r\alpha_1))+f(1-r\alpha_1)\alpha_2)(fr(\alpha_1+\alpha_2)-2)\lambda_{\mathrm{VD}})(\alpha_1\lambda_{\mathrm{VD}}+f(\alpha_1+\alpha_2)(R\theta+\lambda_{\mathrm{VD}}-r\alpha_1\lambda_{\mathrm{VD}}))-2c_p(\alpha_1+\alpha_2)(\alpha_1\lambda_{\mathrm{VD}}+(2f(\alpha_1+\alpha_2)-f^2r(\alpha_1+\alpha_2)^2)(R\theta+\lambda_{\mathrm{VD}}-r\alpha_1\lambda_{\mathrm{VD}}))+2c_f(\alpha_1+\alpha_2)(c_p(\alpha_1+\alpha_2)(1-fr(\alpha_1+\alpha_2))+(1-S_0)(\alpha_1+\alpha_2)(r-fR(2-fr(\alpha_1+\alpha_2))+(\alpha_1(1+f-fr\alpha_1)+f(1-r\alpha_1)\alpha_2)(-2+fr(\alpha_1+\alpha_2))\lambda_{\mathrm{VD}})))$

10. $c_{prl} = \frac{(1-S_0)(2-fr(\alpha_1+\alpha_2))(\alpha_1\lambda_{VD}-c_f(\alpha_1+\alpha_2)+f(\alpha_1+\alpha_2)(R\theta+\lambda_{VD}-r\alpha_1\lambda_{VD}))^2}{2(\alpha_1+\alpha_2)(c_f(\alpha_1+\alpha_2)(fr(\alpha_1+\alpha_2)-1)+\alpha_1\lambda_{VD}+(2f(\alpha_1+\alpha_2)-f^2r(\alpha_1+\alpha_2)^2)(R\theta+\lambda_{VD}-r\alpha_1\lambda_{VD}))}$

11. $c_{psw} = \frac{1}{2(\alpha_1+\alpha_2)(c_f(\alpha_1+\alpha_2)(1-fr(\alpha_1+\alpha_2))-\alpha_1\lambda_{\mathrm{VD}}+(f^2r(\alpha_1+\alpha_2)^2-2f(\alpha_1+\alpha_2))(R\theta+\lambda_{\mathrm{VD}}-r\alpha_1\lambda_{\mathrm{VD}}))}c_f^2(1-S_0)(\alpha_1+\alpha_2)^2(2-fr(\alpha_1+\alpha_2)) + 2c_p(\alpha_1+\alpha_2)(\alpha_1\lambda_{\mathrm{VD}}+(2f(\alpha_1+\alpha_2)-f^2r(\alpha_1+\alpha_2)^2)(R\theta+\lambda_{\mathrm{VD}}-r\alpha_1\lambda_{\mathrm{VD}}))+(1-S_0)(\alpha_1\lambda_{\mathrm{VD}}+f(\alpha_1+\alpha_2)(R\theta+\lambda_{\mathrm{VD}}-r\alpha_1\lambda_{\mathrm{VD}}))(\alpha_1(\lambda_{\mathrm{PF}}+\lambda_{\mathrm{VD}}))-f^2r^2\alpha_1(\alpha_1+\alpha_2)^2\lambda_{\mathrm{VD}}-2(f(\alpha_1+\alpha_2)(R\theta+\lambda_{\mathrm{VD}})+r(\alpha_1+\alpha_2)(2+f^2(\alpha_1+\alpha_2)(R\theta+\lambda_{\mathrm{VD}})+f\alpha_1(2\lambda_{\mathrm{PF}}+3\lambda_{\mathrm{VD}})))+2c_f(\alpha_1+\alpha_2)(c_p(\alpha_1+\alpha_2)(1-fr(\alpha_1+\alpha_2))-(1-S_0)(f^2r^2\alpha_1(\alpha_1+\alpha_2)^2\lambda_{\mathrm{VD}}+2f(\alpha_1+\alpha_2)(R\theta+\lambda_{\mathrm{VD}})+\alpha_1(\lambda_{\mathrm{PF}}+2\lambda_{\mathrm{VD}})-r(\alpha_1+\alpha_2)(1+f^2(\alpha_1+\alpha_2)(R\theta+\lambda_{\mathrm{VD}})+f\alpha_1(\lambda_{\mathrm{PF}}+3\lambda_{\mathrm{VD}}))))$

12. $v_{sw} = max\, \{v_{VD}, v_{PF}\}$, where

$v_{VD} = \frac{1}{(1-S_0)(2-fr(\alpha_1+\alpha_2))(c_f^2(\alpha_1+\alpha_2)^2+(\alpha_1\lambda_{\mathrm{VD}}+f(\alpha_1+\alpha_2)(R\theta+(1-r\alpha_1)\lambda_{\mathrm{VD}}))^2)+2(\alpha_1+\alpha_2)(c_p((f^2r(\alpha_1+\alpha_2)^2-2f(\alpha_1+\alpha_2))(R\theta+(1-r\alpha_1)\lambda_{\mathrm{VD}})-\alpha_1\lambda_{\mathrm{VD}})+c_f(c_p(\alpha_1+\alpha_2)(1-fr(\alpha_1+\alpha_2))-(1-S_0)(2-fr(\alpha_1+\alpha_2))(\alpha_1\lambda_{\mathrm{VD}}+f(\alpha_1+\alpha_2)(R\theta+(1-r\alpha_1)\lambda_{\mathrm{VD}}))))}((1-S_0)(2R\theta - 2\lambda_{PF}-r(f(\alpha_1+\alpha_2)(R\theta-\lambda_{PF})-\alpha_1\lambda_{PF}+r(\alpha_1+\alpha_2)(1+f\alpha_1\lambda_{PF})))(c_f^2(\alpha_1+\alpha_2)^2+(\alpha_1\lambda_{VD}+f(\alpha_1+\alpha_2)(R\theta+\lambda_{VD}-r\alpha_1\lambda_{VD}))^2-2c_f(\alpha_1+\alpha_2)(\alpha_1\lambda_{VD}+f(\alpha_1+\alpha_2)(R\theta+\lambda_{VD}-r\alpha_1\lambda_{VD})))-c_p(\alpha_1+\alpha_2)(\lambda_{PF}-R\theta+r(f(\alpha_1+\alpha_2)(R\theta-\lambda_{PF})-\alpha_1\lambda_{PF}+r(\alpha_1+\alpha_2)(1+f\alpha_1\lambda_{PF})))-2c_p(\alpha_1+\alpha_2)(rR\theta(\alpha_1+\alpha_2)(1-f^2(\alpha_1+\alpha_2)(R\theta-\lambda_{PF})+2f\alpha_1\lambda_{PF})-2\alpha_1\lambda_{PF}\lambda_{VD}+fr^3\alpha_1(\alpha_1+\alpha_2)^2(1+f\alpha_1\lambda_{PF})\lambda_{VD}+r((\alpha_1+\alpha_2)(1-fR\theta((2+f)\alpha_1+f\alpha_2))+((1+f(4+f))\alpha_1^2+2f(2+f)\alpha_1\alpha_2+f^2\alpha_2^2)\lambda_{PF})\lambda_{VD}+2f(\alpha_1+\alpha_2)(R\theta-\lambda_{PF})(R\theta+\lambda_{VD})+R\theta\alpha_1(\lambda_{VD}-\lambda_{PF})-r^2(\alpha_1+\alpha_2)(fR\theta(\alpha_1+\alpha_2)+\alpha_1\lambda_{VD}+f(\alpha_1+\alpha_2+2\alpha_1^2\lambda_{PF})\lambda_{VD}+f^2\alpha_1(\alpha_1+\alpha_2)(R\theta(\lambda_{PF}-\lambda_{VD})+2\lambda_{PF}\lambda_{VD}))))$

$$\begin{aligned}
v_{PF} = {} & \frac{1}{2\left(1 - S_0\right)\left(\alpha_1\lambda_{\mathrm{PF}} - r\left(\alpha_1 + \alpha_2\right)\left(1 + f\alpha_1\lambda_{\mathrm{PF}}\right)\right)\left(c_f\left(\alpha_1 + \alpha_2\right) - \alpha_1\lambda_{\mathrm{VD}} - f\left(\alpha_1 + \alpha_2\right)\left(R\theta + \lambda_{\mathrm{VD}} - r\alpha_1\lambda_{\mathrm{VD}}\right)\right)}(c_f^2(1 - S_0)(\alpha_1 + \alpha_2)^2(R(1 - \theta)(fr(\alpha_1 + \alpha_2) - 2) - 2\lambda_{\mathrm{PF}} + r(\alpha_1 + f\alpha_1 \\
& + f\alpha_2)\lambda_{\mathrm{PF}} - r^2(\alpha_1 + \alpha_2)(1 + f\alpha_1\lambda_{\mathrm{PF}})) - 2c_p(\alpha_1 + \alpha_2)(rR\theta(\alpha_1 + \alpha_2)(1 - f^2(\alpha_1 + \alpha_2)(R(\theta - 1) - \lambda_{\mathrm{PF}}) + 2f\alpha_1\lambda_{\mathrm{PF}}) + fr^3\alpha_1(\alpha_1 + \alpha_2)^2(1 + f\alpha_1\lambda_{\mathrm{PF}})\lambda_{\mathrm{VD}} + r((\alpha_1 + \alpha_2)(1 \\
& + fR(1 - \theta)((2 + f)\alpha_1 + f\alpha_2)) + ((1 + f(4 + f))\alpha_1^2 + 2f(2 + f)\alpha_1\alpha_2 + f^2\alpha_2^2)\lambda_{\mathrm{PF}})\lambda_{\mathrm{VD}} + 2f(\alpha_1 + \alpha_2)(R(\theta - 1) - \lambda_{\mathrm{PF}})(R\theta + \lambda_{\mathrm{VD}}) - \alpha_1(2\lambda_{\mathrm{PF}}\lambda_{\mathrm{VD}} + R(\theta\lambda_{\mathrm{PF}} + \lambda_{\mathrm{VD}} \\
& - \theta\lambda_{\mathrm{VD}})) - r^2(\alpha_1 + \alpha_2)(fR\theta(\alpha_1 + \alpha_2) + \alpha_1\lambda_{\mathrm{VD}} + f(\alpha_1 + \alpha_2 + 2\alpha_1^2\lambda_{\mathrm{PF}})\lambda_{\mathrm{VD}} + f^2\alpha_1(\alpha_1 + \alpha_2)(2\lambda_{\mathrm{PF}}\lambda_{\mathrm{VD}} + R(\theta\lambda_{\mathrm{PF}} + \lambda_{\mathrm{VD}} - \theta\lambda_{\mathrm{VD}})))) - (1 - S_0)(\alpha_1\lambda_{\mathrm{VD}} + f(\alpha_1 + \alpha_2)(R\theta \\
& + \lambda_{\mathrm{VD}} - r\alpha_1\lambda_{\mathrm{VD}}))(-fr^3\alpha_1(\alpha_1 + \alpha_2)^2(1 + f\alpha_1\lambda_{\mathrm{PF}})\lambda_{\mathrm{VD}} + 2f(\alpha_1 + \alpha_2)(R(1 - \theta) + \lambda_{\mathrm{PF}})(R\theta + \lambda_{\mathrm{VD}}) + r(R(\alpha_1 + \alpha_2)(f^2\theta(\alpha_1 + \alpha_2)(R(-1 + \theta) - \lambda_{\mathrm{PF}}) - 2 - f(2 + \theta)\alpha_1\lambda_{\mathrm{PF}}) \\
& + (f\alpha_1(\alpha_1 + \alpha_2)(3R(-1 + \theta) - 4\lambda_{\mathrm{PF}}) + f^2(\alpha_1 + \alpha_2)^2(R(-1 + \theta) - \lambda_{\mathrm{PF}}) - \alpha_1^2\lambda_{\mathrm{PF}})\lambda_{\mathrm{VD}}) + 2\alpha_1(\lambda_{\mathrm{PF}}\lambda_{\mathrm{VD}} + R(\lambda_{\mathrm{PF}} + \lambda_{\mathrm{VD}} - \theta\lambda_{\mathrm{VD}})) + r^2(\alpha_1 + \alpha_2)(fR\theta(\alpha_1 + \alpha_2) + \alpha_1\lambda_{\mathrm{VD}} \\
& + f(\alpha_1 + \alpha_2 + 2\alpha_1^2\lambda_{\mathrm{PF}})\lambda_{\mathrm{VD}} + f^2\alpha_1(\alpha_1 + \alpha_2)(2\lambda_{\mathrm{PF}}\lambda_{\mathrm{VD}} + R(\theta\lambda_{\mathrm{PF}} + \lambda_{\mathrm{VD}} - \theta\lambda_{\mathrm{VD}})))) - 2c_f(\alpha_1 + \alpha_2)(c_p(\alpha_1 + \alpha_2)(r^2(\alpha_1 + \alpha_2) + R(1 - \theta)(1 - fr(\alpha_1 + \alpha_2)) + (1 - r\alpha_1)(1 \\
& - fr(\alpha_1 + \alpha_2))\lambda_{\mathrm{PF}}) + (1 - S_0)(fr^3\alpha_1(\alpha_1 + \alpha_2)^2(1 + f\alpha_1\lambda_{\mathrm{PF}})\lambda_{\mathrm{VD}} + 2f(\alpha_1 + \alpha_2)(R(-1 + \theta) - \lambda_{\mathrm{PF}})(R\theta + \lambda_{\mathrm{VD}}) - \alpha_1(R\lambda_{\mathrm{PF}} + 2(R - R\theta + \lambda_{\mathrm{PF}})\lambda_{\mathrm{VD}}) + r(R(\alpha_1 + \alpha_2)(1 \\
& - f^2\theta(\alpha_1 + \alpha_2)(R(\theta - 1) - \lambda_{\mathrm{PF}}) + f(1 + \theta)\alpha_1\lambda_{\mathrm{PF}}) + (f\alpha_1(\alpha_1 + \alpha_2)(3R(1 - \theta) + 4\lambda_{\mathrm{PF}}) - f^2(\alpha_1 + \alpha_2)^2(R(\theta - 1) - \lambda_{\mathrm{PF}}) + \alpha_1^2\lambda_{\mathrm{PF}})\lambda_{\mathrm{VD}}) - r^2(\alpha_1 + \alpha_2)(fR\theta(\alpha_1 + \alpha_2) \\
& + \alpha_1\lambda_{\mathrm{VD}} + f(\alpha_1 + \alpha_2 + 2\alpha_1^2\lambda_{\mathrm{PF}})\lambda_{\mathrm{VD}} + f^2\alpha_1(\alpha_1 + \alpha_2)(2\lambda_{\mathrm{PF}}\lambda_{\mathrm{VD}} + R(\theta\lambda_{\mathrm{PF}} + \lambda_{\mathrm{VD}} - \theta\lambda_{\mathrm{VD}}))))))
\end{aligned}$$