Software vulnerabilities and bug bounty programs

Carsten Bienz & Steffen Juranek

Discussion Paper No. 18-09
Software vulnerabilities and bug bounty programs

Carsten Bienza, Steffen Juraneka

aNHH Norwegian School of Economics

Abstract

Many software producers employ bug bounty programs that award a prize for the detection of bugs in their software. We analyze in a model with asymmetric information under which conditions a bug bounty program is beneficial for a producer. In our model, a bug bounty program allows producers to perfectly discriminate between different types of bugs, and help to avoid reputation costs of exploited bugs. We find that bug bounty programs are complemented by ex post bug detection efforts. Furthermore, we show that producers can use bug bounty programs to steer hackers attention towards less severe bugs.

Preliminary version, please do not distribute without permission of the authors.

This version: May 31, 2018

Email addresses: carsten.bienz@nhh.no (Carsten Bienz), steffen.juranek@nhh.no (Steffen Juranek)
1. Introduction

How do you avoid being hacked? This question has gained more and more importance in an organizations’ list of issues. Attacks on the software system constitute a significant threat to corporations around the globe. This is a threat to all corporations but it is especially pronounced for software producers. Telang and Wattal (2007) show that vulnerability announcements have a significantly negative effect on a software vendor’s market value.

This problem is further exemplified by an emergence of an active market for hacks in the recent years. Prices on the market show an increasing willingness to pay for software vulnerabilities. For example, market prices for hacks that deliver an attacker control of one of Apple’s iOS devices, so-called zero-day exploits, have risen to above $1 million.\(^1\)

In order to deal with the problem of cybersecurity, software producers try to answer technologically. First, they increase their efforts to improve the coding. Second, they use “fuzzing” methods, i.e., they use random codes to detect anomalies of their programs. Finally, they answer with easier upgrades and increased security features. Microsoft, for example, has begun to force users of its Windows 10 operating system to upgrade in order to immunize their customer’s systems once a security breach has been detected and resolved.

However, technological mechanisms are not sufficient to fully prevent hacks because “it is virtually impossible to design software that is free of vulnerabilities” (Choi et al., 2010, p. 869). Economic incentives can, therefore, be helpful to reduce the damage. One frequently encountered idea is to offer rewards for the disclosure of security vulnerabilities to the software providers, so-called bug bounty program. Many important software producers provide either monetary rewards or cooperate with websites that allow hackers to build up reputational capital. For example, Apple announced their first vulnerability reward

\(^1\)See “iPhone exploit bounty surges to an eye-popping $1.5 million,” Ars Technica, September 30, 2016, available at https://arstechnica.com/information-technology/2016/09/1-5-million-bounty-for-iphone-exploits-is-sure-to-bolster-supply-of-0days/.
program in spring of 2016 and offered to pay up to USD 200,000 for a working exploit\textsuperscript{2}.

Indeed, bug bounty programs appear to be a cost effective way to increase security. Finifter et al. (2013) analyze two existing vulnerability reward programs for the Firefox and Chrome browsers. They find that about 25\% of the detected bugs stem from vulnerabilities disclosed via the program. The total amount of premium paid is by far below the wage costs of full time researchers.

The aim of our study is to analyze under which circumstances it is beneficial for software producers to offer a bug bounty program. We show that there exist two reasons. First, offering ex-ante prizes for the detection of vulnerabilities implies a commitment to pay. That solves an important problem of hackers when trying to sell a vulnerability to the software producer. Without disclosure the software producer cannot assess the bug. However, after disclosure the software producer may use the information to close the vulnerability without paying. Bug bounty programs introduce the legal basis for a commitment to pay for a pre-defined vulnerability. This commitment to pay allows the software producer to perfectly discriminate between type of vulnerabilities, and helps the software producer to avoid reputation costs of exploited bugs. This effect is especially important the more likely and severe high value vulnerabilities are. Furthermore, we show that a bug bounty program is in the low reputation cost case more beneficial if it is complemented by an efficient ex-post detection system; a system that detects vulnerabilities after they are exploited. In contrast, the opposite is true in the high reputation cost case.

Second, we show that a bug bounty program sets ex-ante incentives that allow software producers to direct the attention of the hackers away from high damage vulnerabilities.

Within the small economic literature on cybersecurity, the patching strategy of software bugs has received particular attention. Choi et al. (2010) analyze the optimal patching strategy which is non-trivial and of high importance because not all users update their software in a timely manner and a disclosure allows hackers to reverse-engineer the vulner-

ability. In other words, an update reveals the vulnerability and may lead to exploitation of non-updated versions. Furthermore, Arora et al. (2008) show that the privately optimal patching strategy may be less expeditiously than is socially optimal. August and Tunca (2011) show that this problem can be mitigated by making software producers liable for security vulnerabilities.

We abstract from the optimal patching strategy and rather analyze the role of a bug bounty program on the ability of a software producer to acquire information about vulnerabilities.

2. The model

Players: A hacker (H) works on detecting security gaps in a software manufactured by a software producer (S). If the hacker detects a security gap, she can exploit the gap for up to two periods. There exist two types of security gaps. The exploitation of an L-type gap leads to a monetary transfer of $D_L$ per period from the software producer to the exploiter. An H-type gap implies a monetary transfer of $D_H > D_L$ per period. Furthermore, if a security gap is exploited, it leads to a reputation loss of $R$ for the software producer.

Actions: The hacker searches for security gaps, and finds a gap of type $H$ with probability $p$ and a gap of type $L$ with probability $1 - p$. Once the hacker detects a security gap, he observes its type. The hacker can either monetize herself by selling it on the market or by exploiting it herself, or offer it to the software producer for sale. However, because the hacker cannot easily disclose the information, the software producer does not observe the type, and the two parties are informed asymmetrically. If the hacker would disclose the relevant information, the software producer would receive all the necessary information to close the gap immediately. ³

The software producer can implement two mechanisms to better cope with the asymmetric information. First, the software producer relies always to some extent on ex-post

³In principle, this commitment problem may be solved by repeated interaction. However, because of the large number of individual hackers, there exists barely repeated interaction between a software producer and a particular hacker. That argument also explains the emergence of intermediaries, such as “HackerOne”, which provide an alternative solution to the commitment problem.
detection. After a vulnerability is exploited, the software producer observes the type, and invests resources in closing it. Hence, for different level of damages the software producer invests different amounts in detection. Second, the software producer can implement an *ex-ante bounty program*. With a bounty program the software producer defines different prizes for the two types of vulnerabilities. It commits to pay this prize if the required criteria are met.  

**Timing:** If the software producer implements a bounty program, it defines prizes in $t = 0$. In $t = 1$, the hacker searches for security gaps. If the hacker accepts the respective prize, the game ends. Otherwise, the hacker approaches the software producer notifying her of having found a security gap in $t = 2$. Then, the software producer makes a take-it-or-leave-it offer $B$ to the hacker in exchange for the information. If the hacker accepts the offer, the game ends. If she rejects the offer, she will monetize the gap. The software producer incurs the reputation damage and $D_i$ is transferred from the software producer to the hacker.

In $t = 3$, the software producer invests in detection in order to avoid the second monetary transfer. In particular, the software producer decides upon the detection probability $q$. This choice is costly as the software producer incurs costs of $\frac{1}{2k}q^2$. The exploitation of the gap leads to revenues for the hacker of $D_i$ in $t = 2$, and the same amount in $t = 3$ if the software producer is unable to close the gap. The software producer incurs the damage $D_i$ per period plus a one-time reputation cost $R$. Figure 1 summarizes the timing.

```
<table>
<thead>
<tr>
<th>t=0</th>
<th>t=1</th>
<th>t=2</th>
<th>t=3</th>
</tr>
</thead>
<tbody>
<tr>
<td>S sets prizes for gaps</td>
<td>Hacker (invests and) finds a gap</td>
<td>H demands the prize <strong>or</strong> H approaches S, S makes a take-it-or leave-it-offer</td>
<td>If H denies the offer, S invests in detection</td>
</tr>
</tbody>
</table>
```

Figure 1: Timing

---

4In order to credibly commit, the software producer can predefine the ability of a vulnerability and cover this by a contract. Additionally, it can also outsource the decision authority to an external committee.
2.1. Exogenous hacker effort

2.1.1. No bounty program

Because our aim is to analyze the decision of introducing a bounty program, we start our analysis with the benchmark cases of no bounty program. In this case, the game starts with stage $t = 1$. We solve the respective game by backward induction. In case of a failure of negotiations, the firm observes the type of the gap and invests in period $t = 3$ in detection. The software producer maximizes its expected payoff:

$$\max_{q_i} \Pi^S = -D_i(1 - q_i) - \frac{1}{2k}q_i^2 - D_i - R.$$ 

Consequently, the software producer invests $q_i^* = kD_i$, the software producer’s expected payoff equals $\Pi^S = -D_i(2 - kD_i) - R$, whereas the hacker expects a payoff of $\Pi^H = D_i(2 - kD_i)$.

Both players take this decision into account in period $t = 2$. The software producer makes the hacker an offer $B$. The hacker can either accept or reject the offer. If she rejects the offer, both proceed to stage 3. Note that a rejection does not result in a zero-sum game because the software producer incurs the reputation costs $R$ and the detection costs $\frac{1}{2}kD_i^2$.

Because the software producer does not observe the type of vulnerability, it cannot discriminate with different offer. Therefore, the software producer has three options. First, the software producer makes an offer that the hacker accepts for both types of gaps. The minimal offer that $H$-type (and $L$-type) hackers accept equals $B_2 = D_H(2 - kD_H)$ and the firm’s payoff equals $-B_2$. Second, the software producer makes an offer that only hackers with an $L$-type gap accept. In this case, the offer equals $B_1 = D_L(2 - kD_L)$, and the software producer’s expected payoff follows as $-(1 - p)B_1 - p\left(D_H(2 - kD_H) + R\right)$. Third, the firm can decide to make no offer. The firm’s payoff is then given by $-(1 - p)\left(D_L(2 - kD_L)\right) - p\left(D_H(2 - kD_H)\right) - R$. 

6
The optimal offer depends on the different parameters. Solving for the optimal offer allows us to state Lemma 1.

**Lemma 1. Without a bounty program**

1. The software producer always makes an offer.
2. The software producer offers $D_H(2 - kD_H)$ and pays off both types if
   \[ R \geq \tilde{R} \equiv \frac{1 - p}{p} (D_H(2 - kD_H) - D_L(2 - kD_L)) - k \frac{D_H^2}{2}. \]

   The software producer’s payoff equals
   \[ \Pi^{S}_{nobody} = -D_H(2 - kD_H). \]

3. Otherwise, the software producer bids $D_L(2 - kD_L)$ and only hackers with a low type gap accept. The software producer’s expected payoff equals
   \[ \Pi^{S}_{nobody} = -(1 - p)D_L(2 - kD_L) - p \left( D_H(2 - k \frac{D_H}{2}) + R \right). \]

**Proof.** See the Appendix.

It is always optimal for the software producer to pay off at least the low type gaps in order to avoid the reputation and detection costs. The decision whether to pay off the high types, too, depends on the trade off between avoiding the reputation costs, and the higher pay to the low types. The software producer pays off both types, the higher $p$ and $R$ are, and the lower $D_H - D_L$ is. If $p$ is high, the low type gaps are relatively unimportant, and the hacker profits from the high payment only in few cases. A higher $R$ increases the direct costs of not paying off the high types. Furthermore, because the software producer can invest in detection, the outside option of the hacker decreases. The decrease leads to a lower offer and a gain for the software producer. Finally, a smaller difference between the damage of the two types decreases the amount overpaid to the low types.

Because, we treat the hacker as passive in this baseline case, Lemma 1 summarizes the equilibrium if the software producer does not employ a bounty program.

2.1.2. Bounty program

With a bounty program the games starts already in $t = 0$. In the bounty program, the firm commits to pay a pre-defined prize for the two types, $P_H$ and $P_L$. After finding
a gap in \( t = 1 \), the hacker can demand the respective prize. In case the hacker rejects the prize, she approach the firm again in \( t = 2 \). If negotiations fail she can sell the gap on the market, and the firm invests in detection. Solving the game allows us to state Lemma 2.

**Lemma 2.** With a bounty program

1. the software producer announce the prize \( P_H = D_H (2 - kD_H) \) for H-type gaps.
2. the software producer announce the prize \( P_L = D_L (2 - kD_L) \) to the L-type gaps.
3. the software producer’s payoff equals \( \Pi_{bounty}^F = - (1-p)D_L (2 - kD_L) - pD_H (2 - kD_H) - F \).

The hacker always accepts the prize.

It follows straightforwardly that none of the players has an incentive to deviate from this equilibrium. The prize ensures a hacker with a H-type vulnerability has no incentive to reject the prize as she will not be able to gain more in the later stages. Consequently, the software producer knows that a hacker rejecting the offer has a L-type hack. Consequently, it would offer \( P_L \) in stage 2, which the hacker would accept. Hence, no one has an incentive to deviate from the equilibrium described in Lemma 2.

The bounty program allows the software producer to perfectly differentiate between the two types of vulnerabilities. By committing to pay the pre-defined prize, the hackers will reveal the type of their vulnerability, and the information asymmetry gets resolved. Consequently, the software producer can pay each type of hacker the required amount.

**2.1.3. Comparison**

Comparing the payoff in Lemma 2 with the payoff in Lemma 1 allows us to state Proposition 1.

**Proposition 1.** The incremental payoff of a bounty program is given by

\[
\pi^S = \min \left( p \left( R + k \frac{D_H^2}{2} \right), (1-p) \left( D_H (2 - kD_H) - D_L (2 - kD_L) \right) \right) - F.
\]

The advantages of the bounty program depend on the parameter setting. In the low reputation cost case, the software producer saves with a bounty program the reputation costs and the resources that are spend on detection. In the high reputation cost case, the
bounty program allows the software producer to perfectly differentiate between the two types, rather than paying off both types.

Hence, the decision whether the software producer employs a bounty system depends on the fixed costs of the system. We define the critical fixed costs such that the incremental payoff of the system just equals zero, i.e.,

\[ F^c = \min \left( p \left( R + k \frac{D_H^2}{2} \right), (1 - p) (D_H (2 - kD_H) - D_L (2 - kD_L)) \right). \]

From this definition, we derive Proposition 2.

**Proposition 2.**

- If \( R < \tilde{R} \), then the critical fixed costs \( F^c \) increase in \( k, D_H, R \) and \( p \).
- If \( R \geq \tilde{R} \), then the critical fixed costs \( F^c \) increase in \( D_H \) but decrease in \( p, k \) and \( D_L \).

Figure 2 illustrates the comparative static effects. An increase of \( D_H \) unambiguously decreases \( F^c \) because \( D_H \) increases the costs for detection as well as the excess payment for L-type gaps in the high reputation cost case. In the same line of reasoning, the damage of the L-type gaps does not influence \( F^c \) in the low reputation cost case because with and without a system, the software producer always pays the same amount for L-type gaps. However, in the high reputation cost case, \( F^c \) decreases in \( D_L \) because a higher low-type damage decreases the benefits of differentiation between the two types.

A higher share of H-type gaps (higher \( p \)) increases the payoff of the bounty program in the low reputation cost case because it avoids the reputation and detection costs more often. Furthermore, the payoff in the high reputation cost case decrease because differentiation becomes less important the lower the share of L-type gaps is. Furthermore, an additional payoff decreasing effect stems from the influence of \( p \) on the likelihood of the two cases. An increase in \( p \) makes the low reputation cost case less likely.

Finally, the detection cost parameter \( k \) has a similar effect. The lower the marginal detection costs (higher \( k \)), the more beneficial is the bounty system in the low reputation cost case. The contrary is true for the high reputation cost case. If \( k \) is high, the software producer spends more on detection. These costs are avoided with a bounty program in the low reputation cost case. However, an increase in \( k \) also decreases the difference
in the payments for L- and H-type gaps, which makes a differentiation less appealing. Additionally, a higher $k$ also has an effect on the probability of the two cases; it makes the low reputation cost case more likely. The lower left figure in Figure 2 illustrates the observations graphically. From this observation it become obvious that a bounty program complements the detection system in the low reputation cost case, whereas the bounty program and the detection are partial substitutes in the high reputation case.

The latter observation has an important consequence on the design of the optimal security system of a software producer. If the software producer has only low reputation costs to fear, a bounty program complements investments in a detection system. On the contrary, if a firm is prone to high reputation costs, a bounty system will substitute for higher investments in a detection system.
2.2. Endogenous hacker effort

2.2.1. No bounty program

In this section, we include an active hacker in period \( t = 1 \) and allow the hacker to choose the probability \( p \) endogenously. The hacker can search more intensely and increase her chances to find the high type gap. However, she incurs costs of \( \frac{1}{2} cp^2 \). We assume that the hacker’s effort is observable but that the outcome remains unknown. Hence, the software producer still observes the probability for a low or high damage. Finally, for simplicity we assume that the costs are between \( \Delta \leq c \leq 2\Delta \).

The stages \( t = 3 \) and \( t = 2 \) are not directly affected by making the hacker’s effort endogenous, and remain unchanged; Lemma 1 continues to hold. We can use this result in
order to analyze the optimal behavior of the hacker. Lemma 1 tells us that the software producer pays off both types if $R \geq \hat{R}$. This can be translated to $p \geq \tilde{p} \equiv \frac{\Delta}{\Delta + R + k \frac{D_H}{2}}$, with $\Delta = D_H(2 - k \frac{D_H}{2}) - D_L(2 - k \frac{D_L}{2})$. Consequently, a hacker never picks a $p$ above $\tilde{p}$.

The hacker maximizes the expected payoff

$$
\max_p \quad \Pi^H_{\text{no bounty}}(p) = pD_H(2 - kD_H) + (1 - p)D_L(2 - kD_L) - \frac{1}{2}cp^2,
$$

as long as the profit with choosing $\tilde{p}$ and achieving the high offer for sure instead is not higher. Following from this setup we can summarize:

**Lemma 3. Without a bounty program**

1. The hacker picks $\tilde{p} = \frac{\Delta}{\Delta + R + k \frac{D_H}{2}}$ if $R \geq \hat{R} \equiv c\sqrt{\frac{\Delta}{2c - \Delta}} - \Delta - k \frac{D_H}{2}$.

2. Otherwise, the hacker picks $p^* = \frac{\Delta}{c}$. The expected payoffs follow as $\Pi^H_{\text{no bounty}}(\tilde{p}) = \frac{\Delta^2}{2(\Delta + R + k \frac{D_H}{2})}$ and $\Pi^S_{\text{no bounty}}(p^*) = -D_H(2 - kD_H)$.

**Proof.** See the Appendix.

### 2.3. Bounty program

By introducing a bounty program and defining prizes in $t = 0$, the software producer can influence the behavior of the hacker. Therefore, the actions of the hacker depend on the prizes defined in $t = 1$.

Assuming that it is always optimal for the hacker to accept the prizes, she maximizes her expected payoff

$$
\max_p \quad \Pi^H_{\text{bounty}}(p) = pP_H + (1 - p)P_L - \frac{1}{2}cp^2.
$$

Consequently, the hacker picks $p = \frac{P_H - P_L}{c}$.

The software producer has no leeway in determining the high type prize $P_H$ because a hacker always gets a payoff of $D_H(2 - kD_H)$ for an H-type hack. Hence, the firm has to offer that amount as a prize, i.e., $P_H = D_H(2 - kD_H)$. 
Therefore, the maximization problem of the firm becomes

\[
\max_{P_L} \Pi_{\text{bounty}}^F = -\frac{D_H(2 - kD_H) - P_L}{c} D_H(2 - kD_H) - \left(1 - \frac{D_H(2 - kD_H) - P_L}{c}\right) P_L - F.
\]

Solving the maximization problem let’s us state

**Lemma 4.** With a bounty program

1. the firm announce the prize \( P_H = D_H(2 - kD_H) \) for H-type gaps,
2. the firm announce the prize \( P_L = D_H(2 - kD_H) - \frac{c}{2} \) to the L-type gaps,
3. the hacker picks \( p^*_\text{bounty} = \frac{1}{2} \) and always accepts the prize,
4. the firm’s payoff equals \( \Pi_{\text{bounty}}^F = -D_H(2 - kD_H) + \frac{c}{4} - F \).

Clearly, a hacker with a H-type gap cannot gain more than \( P_H \) by rejecting the prize. For a hacker with an L-type hack to accept the prize it has to hold that \( P_L \geq D_L(2 - kD_L) \). This condition is always fulfilled for interior solutions of \( p \), i.e., if \( c \leq 2\Delta \).5 Interestingly, this condition also implies that the software producer pays the hacker a premium for the L-type hacks, i.e., \( P_L > D_L(2 - kD_L) \). By doing so, the software producer pays the hackers for searching less intensely for the high value hacks. The bounty program allows the software producer to draw the hackers attention to the less severe areas.

**2.3.1. Comparison**

Comparing Lemma 3 and 4 allows us to state Proposition 3.

**Proposition 3.** The incremental payoff of a bounty program is given by

\[
\pi = \begin{cases} 
\frac{c}{4} - F & \text{if } R \geq \hat{R} \\
\frac{\Delta}{c} \left(R + kD_H^2\right) + \left(\frac{\Delta}{c} - \frac{1}{2}\right) \Delta - \frac{1}{2} \left(\Delta - \frac{c}{2}\right) - \frac{c}{4} - F + \frac{\Delta}{c} \left(R + kD_H^2 + \Delta - c\right) & \text{if } R < \hat{R}
\end{cases}
\]

The intuition behind the result is pretty similar to the previous section. If \( R \geq \hat{R} \), the bounty program helps to differentiate between the two types that are equally likely. If \( R < \hat{R} \) the bounty program saves reputation costs and the resources spent on detection. Additionally, the firm has to pay the high prize less often because the hacker invests less. However, this comes at the cost of higher payments to the hacker for low-type hacks.

This result let us also state the comparative static effects.

5Note that larger costs \( c \) lead to the corner solution \( p = 0 \). Then the software producer knows the type of the hack, and pays \( P_L = D_L(2 - kD_L) \).
Proposition 4.

- If $R < \bar{R}$, then the critical fixed costs $F^c$ increase in $k$, $D_H$, $D_L$ and $R$ but decreases in $c$.
- If $R \geq \bar{R}$, then the critical fixed costs $F^c$ increase only in $c$.

Figure 3 illustrates the comparative static effects. The main difference between endogenous and exogenous hacker choices is that the difference between the high and low payment depends purely on the costs of effort $c$. The higher the costs $c$ the more difficult it is for the hacker to find the H-type gaps. Hence, the software producers have to disincentives the hackers less for searching for H-type gaps. Hence, the benefit of the bounty program in the high reputation costs case is especially pronounced if $c$.

In the low reputation costs case, the H-type damage increases the profitability of the bounty program, whereas the L-type damage decreases it. This is because the software producer saves the detection costs $k\frac{D_H^2}{2}$ but also because the larger the difference between $D_H$ and $D_L$, the harder the hacker search for H-type gaps without the bounty program, and the bounty program becomes especially attractive to manage the hackers activities. The hackers costs $c$ decrease the profitability of the bounty program in the low reputation cost case because high costs imply a low share of H-type gaps, and less detection costs that can be saved with the bounty program. Finally, in line with the exogenous hacker choice, we also observe the complementarity between ex-post detection and the bounty program here as well.

3. Discussion and conclusion

Our preliminary results show two benefits of a bug bounty program. First, pre-defining payments for vulnerabilities in bug bounty programs solve the commitment problem and allow perfect differentiation between types of vulnerabilities. Naturally, this effect is especially important the more severe high value vulnerabilities are.
In the low reputation cost case, without the bounty program the software producer would pay only for the low damage vulnerabilities and incur damage for the high damage gaps. Hence, the bug bounty program saves the software producer potential reputation and ex-post detection costs, and consequently is more beneficial if the high damage gaps are more likely, and if the software producer operates a more efficient ex-post detection system.

In the high reputation cost case, the software producer would pay the same (high) price for all types of hacks without a bounty program. Hence, because the bounty program allows perfect discrimination, it becomes more efficient the more likely low damage gaps and the less efficient the ex-post detection system is. The latter effect stems from the fact
that the consequences of both types of vulnerabilities become more similar with a more efficient ex-post detection system.

Second, we show that a bug bounty program sets ex-ante incentives that allow software producers to direct the attention of the hackers away from high damage vulnerabilities.

It is obvious that especially the latter effect may cause additional unintended consequences such as entry. However, up to now we ignore the effect of competing hackers. Competing hackers matter for our set-up because more than one hacker may find the same vulnerability. This may allow the software producer to push down the prices for the vulnerabilities. A straight-forward extension of our current set-up is, therefore, to allow for competition and entry, and to analyze the interaction of competition with the bug bounty program.

Furthermore, we currently assume that the damage is a monetary transfer from the software producer to the hacker. However, there may be empirically relevant cases for which this is not the case. This is because the software producer is typically not fully liable for damage that occurs via their product. This is another topic that we plan to discuss with a more complete model.
References


Appendix

Proof of Lemma 1

Step 1: Alternative 2 dominates 3 if

\[(1 - p)B_1 + p \left( D_H(2 - kD_H) + R \right) < (1 - p) \left( D_L(2 - kD_L) \right) + p \left( D_H(2 - kD_H) \right) + R \]

which translates to

\[(1 - p) \left( k \frac{D_L}{2} + R \right) > 0 \]

Hence, the firm will at least always make an offer that the hacker accepts L-type gaps.

Step 2: Alternative 1 dominates 2 if

\[B_2 < (1 - p)B_1 + p \left( D_H(2 - kD_H) \right) \]

which translates to

\[(1 - p) \left( D_H(2 - kD_H) - D_L(2 - kD_L) \right) < p \left( k \frac{D_L^2}{2} + R \right) \]

Proof of Lemma 3

Maximizing \(\Pi^H_{\text{no bounty}}(p) = pD_H(2 - kD_H) + (1 - p)D_L(2 - kD_L) - \frac{1}{2}c p^2\) gives \(p^* = \frac{\Delta}{c}\)

and the associated expected payoff is given by \(\Pi^H_{\text{no bounty}}(p^*) = \frac{\Delta^2}{2c} + D_L(2 - kD_L)\). In contrast, the profit with \(\hat{p}\) is given by \(\Pi^H_{\text{no bounty}}(\hat{p}) = \Delta + D_L(2 - kD_L) - \frac{c\Delta^2}{2 \left( \Delta + k \frac{D_H^2}{2} \right)^2}\).

It holds that \(\Pi^H_{\text{no bounty}}(\hat{p}) > \Pi^H_{\text{no bounty}}(p^*)\) if \(R \geq \hat{R} \equiv \frac{\Delta^2}{\sqrt{2c-\Delta}} - \Delta - k \frac{D_H^2}{2}\). Hence, if \(R \geq \hat{R}\) the hacker picks \(\hat{p}\), and the firm pays the hacker \(D_H(2 - kD_H)\)

18