

Popularity of Free Software generating bug exploitation?

Iñaki Silanes

University of the Basque Country (UPV/EHU)
Donostia, Spain

November 23, 2006

1 Introduction

It is often said (by FLOSS-skeptics), that Free Software has less exploited bugs than the Proprietary Software because it is less popular. They argue that, since less people uses FLOSS, the crackers are less inclined to waste their time exploiting the bugs it could have. The greater user base of the proprietary software would also, in their words, make bugs more prominent, and their exploits spread faster. The corollary of this theory would be that popularization of FLOSS applications (e.g. Firefox), would lead to an increase in the number of bugs discovered and exploited, eventually reaching a proprietary-like state (e.g. “*Firefox will have as many bugs as IE, when Firefox is as popular as IE*”).

The objective of this paper is to mathematically demonstrate the utter nonsense of this theory. Specifically, I will argue that an increase in community size benefits a FLOSS project in at least 3 ways:

1. Faster development
2. Shorter average life of open bugs
3. Shorter average life of exploited bugs

2 Propositions and derivation

Imagine we have a FLOSS project \mathcal{P} . Let us postulate that the developers will release new versions with a period \mathbf{T} , each release incorporating \mathbf{G} new bugs. For simplicity’s sake, let us also consider that each new version will be released when all bugs from the previous release have been patched.

I will also assume that there will be a bug patching rate, \mathbf{P} , dependant upon the size of the community, \mathbf{U} , using, testing and contributing to \mathcal{P} . If we assume this dependence to be a *proportionality*, we get Eq. 1, where K_p is a constant that can be taken as the “patching efficiency” of the community.

$$P = K_p U \tag{1}$$

Since P is the rate at which the number of open bugs, β , are patched, the time-dependence of β can be readily calculated from Eq. 1, and it is done in Eq. 2.

$$\begin{aligned}
\frac{d\beta}{dt} &= -K_p U \\
\int_G^\beta d\beta &= -K_p U \int_0^t dt \\
\beta &= G - K_p U t
\end{aligned}
\tag{2}$$

The period of time between \mathcal{P} version releases (T), can be derived from the simple Eq. 2. Solving for $\beta = 0$ we get Eq. 3.

$$\begin{aligned}
0 &= G - K_p U T \\
T &= \frac{G}{K_p U}
\end{aligned}
\tag{3}$$

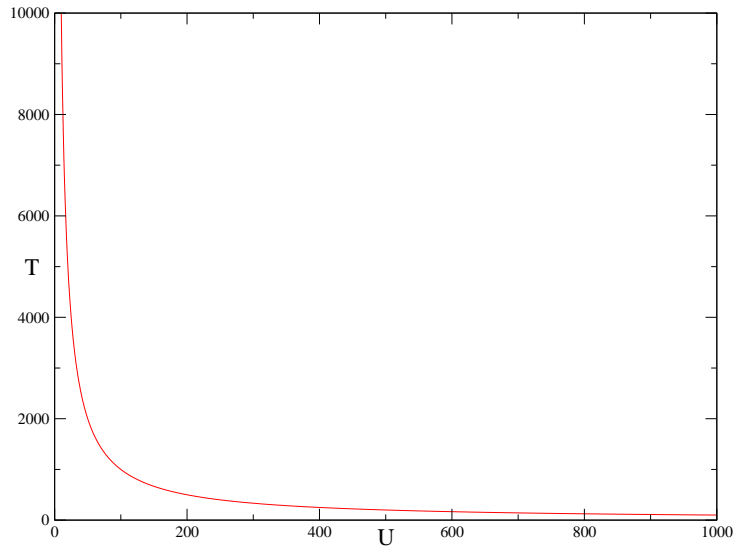


Figure 1: Inter-release time (T) *vs.* amount of users in community (U), according to Eq. 3, for constant $K_p = 0.01$ and $G = 1000$.

The interpretation of Eq. 3 is straightforward: the more bugs in a release (G), the longer it will take to make the next release. Conversely, the broader the user base (U), and/or faster patch rate per user (K_p), the shorter inter-release time, or, in other words:

Increasing the size of the user base of a FLOSS program speeds up its development.

2.1 Average lifetime of bugs

In the time period between t and $t+dt$, $(-d\beta/dt)dt$ bugs are patched (patching speed times time interval). Recall that $d\beta/dt$ is negative). Their age (how long they have been open) would be t . The average age of bugs would, then, be the sum of the ages of all bugs, times the number of bugs of *that* age, and divided by the total number of bugs. Calling τ the average lifetime of bugs, we can make the derivation in Eq. 4.

$$\begin{aligned}
 \tau &= \frac{\int t \left(\frac{-d\beta}{dt} \right) dt}{\int d\beta} \\
 \tau &= \frac{1}{G} \int_0^T t (K_p U) dt \\
 \tau &= \frac{K_p U}{G} \left[\frac{t^2}{2} \right]_0^T \\
 \tau &= \frac{1}{T} \frac{T^2}{2} \\
 \tau &= \frac{T}{2}
 \end{aligned} \tag{4}$$

What Eq. 4 tells us is that the average lifetime of bugs (how long a bug stays unpatched) is always one half of the inter-release time (T). See Figure 1 for the evolution of T with U . This means that τ , just as T , is inversely proportional to the size of the community or, in other words:

Increasing the size of the user base of a FLOSS program reduces the average time bugs stay unpatched.

2.2 Fraction of bugs exploited before being patched

If we were to calculate the fraction of bugs that gets actually exploited during its lifetime, we would have to define a “bug exploitation rate”, which I’ll propose can be proportional to the size of the community (U). The rationale behind it is that the more people using \mathcal{P} , the more crackers among them, and also the more “interesting” for a cracker to exploit a bug. This rate would also be proportional to the number of open *and* still unexploited bugs, β_{ou} . If this were the case, Eq. 5 would hold.

$$\frac{d\beta_x}{dt} = K_x U \beta_{ou}(t) \tag{5}$$

To solve Eq. 5, one has to take into account the definitions in Eqs. 6 – 8, where β_{ox} and β_{px} stand for the amount of exploited bugs that are, respectively, still open and already patched.

$$\beta = \beta_{ou} + \beta_{ox} \tag{6}$$

$$\beta_x = \beta_{ox} + \beta_{px} \tag{7}$$

$$\alpha = \frac{\beta_{ox}}{\beta} \tag{8}$$

Eqs. 6 – 8 help calculate the evolution of both β_{ou} and β_{ox} with time. For β_{px} we have Eq. 9.

$$\frac{d\beta_{px}}{dt} = \overbrace{\frac{d\beta}{dt}}^{\text{patching speed}} \underbrace{\frac{\beta_{ox}}{\beta}}_{\text{exploited fraction}} = K_p U \frac{\beta_{ox}}{\beta} \quad (9)$$

For solving Eq. 9, we need $\beta_{ox}(t)$ beforehand. To calculate it, we make the derivation in Eq. 10.

$$\begin{aligned} \frac{d\beta_{ox}}{dt} &= \overbrace{K_x U \beta_{ou}}^{\text{exploited}} - \overbrace{\frac{d\beta}{dt} \frac{\beta_{ox}}{\beta}}^{\text{patched}} \\ \frac{d\beta_{ox}}{dt} &= \frac{d(\alpha\beta)}{dt} = \alpha \frac{d\beta}{dt} + \beta \frac{d\alpha}{dt} \\ \alpha \frac{d\beta}{dt} + \beta \frac{d\alpha}{dt} &= K_x U \beta_{ou} + \frac{d\beta}{dt} \frac{\beta_{ox}}{\beta} \\ \alpha \frac{d\beta}{dt} + \beta \frac{d\alpha}{dt} &= K_x U (1 - \alpha) \beta + \alpha \frac{d\beta}{dt} \\ \frac{d\alpha}{dt} &= K_x U (1 - \alpha) \\ \frac{d\alpha}{1 - \alpha} &= K_x U dt \\ \int_0^\alpha \frac{d\alpha}{1 - \alpha} &= \int_0^t K_x U dt \\ -\ln(1 - \alpha) &= K_x U t \\ \alpha &= 1 - e^{-K_x U t} \end{aligned} \quad (10)$$

Joining Eqs. 2, 8 and 10, we get an expression for β_{ox} , namely that in Eq. 11.

$$\beta_{ox} = \alpha(t)\beta(t) = (1 - e^{-K_x U t}) (G - K_p U t) \quad (11)$$

We can now develop Eq. 9, including the formula in Eq. 11, which gives us Eq. 12.

$$\begin{aligned} \frac{d\beta_{px}}{dt} &= -\frac{d\beta}{dt} \frac{\beta_{ox}}{\beta} = K_p U (1 - e^{-K_x U t}) \\ \int_0^{\beta_{px}} d\beta_{px} &= K_p U \int_0^t (1 - e^{-K_x U t}) dt \\ \beta_{px} &= K_p U \left[t + \frac{e^{-K_x U t}}{K_x U} \right]_0^t \\ \beta_{px} &= K_p U t - \frac{K_p}{K_x} (1 - e^{-K_x U t}) \end{aligned} \quad (12)$$

From Eqs. 7, 11 and 12, we can calculate the total (open and patched) amount of exploited bugs at any given moment (β_x), and dividing by the total amount of bugs generated (G), obtain the fraction of bugs that get exploited ever. This is given in Eq. 14, where γ is defined as in Eq. 13.

$$\gamma = \frac{K_p}{K_x G} \quad (13)$$

$$\begin{aligned}
\beta_x &= \beta_{ox} + \beta_{px} \\
\beta_x(t) &= (G - K_p Ut) (1 - e^{-K_x Ut}) + K_p Ut - \frac{K_p}{K_x} (1 - e^{-K_x Ut}) \\
\beta_x(t) &= G - K_p Ut - G e^{-K_x Ut} + K_p U t e^{-K_x Ut} + K_p Ut - \frac{K_p}{K_x} - \frac{K_p}{K_x} e^{-K_x Ut} \\
\beta_x(t) &= G - G e^{-K_x Ut} + K_p U t e^{-K_x Ut} - \frac{K_p}{K_x} - \frac{K_p}{K_x} e^{-K_x Ut} \\
\beta_x(t) &= G - \frac{K_p}{K_x} + \left(K_p Ut - G - \frac{K_p}{K_x} \right) e^{-K_x Ut} \\
\frac{\beta_x(t)}{G} &= 1 - \frac{K_p}{K_x G} + \left(\frac{K_p}{K_x G} - 1 + \frac{K_p Ut}{G} \right) e^{-K_x Ut} \\
\frac{\beta_x(t)}{G} &= 1 - \gamma + \left(\gamma - 1 + \frac{K_p Ut}{G} \right) e^{-K_x Ut} \tag{14}
\end{aligned}$$

We would now want to know the total fraction of bugs that were exploited by the time all bugs have been patched, that is $F_x = \beta_x(T)/G$. For that, I will make use of Eqs. 3 and 14, and obtain Eq. 15.

$$\begin{aligned}
F_x &= 1 - \gamma + \left(\gamma - 1 + \frac{K_p UT}{G} \right) e^{-K_x UT} \\
F_x &= 1 - \gamma + \left(\gamma - 1 + \frac{K_p U \frac{G}{K_p U}}{G} \right) e^{-K_x U \frac{G}{K_p U}} \\
F_x &= 1 - \gamma + (\gamma - 1 + 1) e^{-K_x \frac{G}{K_p}} \\
F_x &= 1 - \gamma + \gamma e^{-1/\gamma} \tag{15}
\end{aligned}$$

From Eq. 15 it follows that the fraction of the initial G bugs that end up ever being exploited is **independent of U** , and solely depends on γ , that is, the $K_p/K_x G$ ratio. This makes sense, since I have proposed that the vaster the community of users of program \mathcal{P} , the faster bugs will be patched (Eq. 1), but **also** the faster bugs will get exploited (Eq. 5).

A sanity check can be made on Eq. 15, testing the upper and lower limits of γ , that is, when the initial bug number (G), and/or the effectiveness of the crackers to do their nasty job (K_x), is much greater ($\gamma \rightarrow 0$) and smaller ($\gamma \rightarrow \infty$) than the effectiveness of the community to patch the bugs (K_p). This sanity check is presented in Eq. 16, and confirms the intuitive idea that for infinitely efficient patching user community, all bugs would be patched **before** ever getting exploited. In the opposite case, all bugs would be exploited at some point in their long lives. The F_x *vs.* γ dependency is also plotted in Figure 2.

$$\begin{aligned}
\lim_{\gamma \rightarrow 0} F_x(\gamma) &= 1 \\
\lim_{\gamma \rightarrow \infty} F_x(\gamma) &= 0 \tag{16}
\end{aligned}$$

2.3 Average life of exploited bugs

I am now concerned with how long open exploited bugs stay open. I will start assuming that the exploited bugs are just as likely to be patched as unexploited ones. Notice that this overestimates

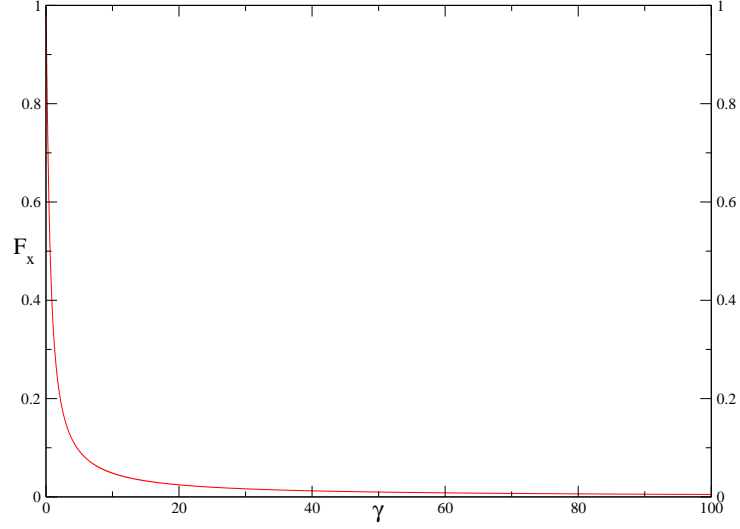


Figure 2: Evolution of F_x (see Eq. 15) with γ (see Eq 13).

how long exploited bugs stay unpatched, because usually exploited bugs are paid more attention than known but unexploited ones.

Let us write down the magnitudes we will need for further discussion:

1. Bugs exploited between t and $t + dt$ (Eq. 5):

$$expl(t) = K_x U \beta_{ou} dt = K_x U (1 - \alpha) \beta dt$$

2. Fraction of total open bugs that are exploited between t and $t + dt$:

$$frac(t) = \frac{expl(t)}{\beta(t)} = K_x U (1 - \alpha) dt$$

3. Fraction of open bugs at t' that corresponds to the ones exploited at t :

$$frac(t') = frac(t)$$

4. Total exploited bugs of exploitation age $\Delta t = t' - t$, at t' :

$$to(\Delta t) = frac(t') \beta(t')$$

5. Number of exploited bugs patched at t' (Eqs. 1 and 8):

$$pa(t') = K_p U \alpha(t') dt'$$

6. Fraction of total open bugs that represent the exploited ones that are patched at t' :

$$frap(t') = \frac{pa(t')}{\beta(t')}$$

7. Bugs of exploited age Δt patched at t' :

$$p(\Delta t) = to(\Delta t) frap(t')$$

$$\begin{aligned}
p(\Delta t) &= to(\Delta t) \frac{pa(t')}{\beta(t')} \\
p(\Delta t) &= \frac{expl(t)}{\beta(t)} \beta(t') \frac{pa(t')}{\beta(t')} = \frac{expl(t)}{\beta(t)} pa(t') \\
p(\Delta t) &= K_x K_p U^2 (1 - \alpha(t)) dt \alpha(t') dt'
\end{aligned}$$

We can integrate the product of $p(\Delta t)$ above by its age (Δt), and divide by the total bugs ever exploited ($\beta_{px}(T)$), to obtain the average exploitation time of the bugs that were ever exploited. The corresponding derivation is given in Eq. 17, where τ and F_x are the same ones as in Eqs. 4 and 15, respectively.

$$\begin{aligned}
\tau_x &= \frac{\int \Delta t p(\Delta t)}{\beta_{px}(T)} \\
\tau_x &= \frac{\int \int (t' - t) K_x K_p U^2 (1 - \alpha(t)) dt \alpha(t') dt'}{\beta_{px}(T)} \\
\tau_x &= \frac{K_x K_p U^2 \int \int (t' - t) (e^{-K_x U t}) dt (1 - e^{-K_x U t'}) dt'}{K_p U T - \frac{K_p}{K_x} (1 - e^{-K_x U T})} \\
\tau_x &= \frac{K_x K_p U^2 \int_{t=0}^{t=T} e^{-K_x U t} dt \int_{t'=t}^{t'=T} (t' - t) (1 - e^{-K_x U t'}) dt'}{G - \frac{K_p}{K_x} (1 - e^{-1/\gamma})} \\
\tau_x &= \frac{K_p \left(1 + e^{G K_x / K_p} \left(\frac{G K_x}{K_p} - 1 \right) \right)^2 e^{-2 G K_x / K_p}}{2 K_x^2 U} \\
\tau_x &= \frac{G \left(1 - \frac{K_p}{G K_x} (1 - e^{-1/\gamma}) \right)}{2} \\
\tau_x &= \frac{T \gamma^2 \left(1 + \left(\frac{1}{\gamma} - 1 \right) e^{1/\gamma} \right)^2 e^{-2/\gamma}}{(1 - \gamma (1 - e^{-1/\gamma}))} \\
\tau_x &= F_x \tau
\end{aligned} \tag{17}$$

As can be seen, the average time exploited bugs stay unpatched is proportional to the inter-release period (T). The proportionality constant (F_x), depends solely on the previously defined γ , and is thus **independent** of the size of the community (U). For small γ ($K_p \ll \ll K_x G$), F_x approaches unity, so that $\tau_x \sim \tau = T/2$. Conversely, the larger γ (K_p increases with respect to $K_x G$), the smaller τ_x , or, in other words, the shorter the exploitation time, even with respect to T . Note, however, that T itself is also affected by the size of the community (U), as given by Eq. 3.

We can plot τ_x *versus* all the parameters it depends upon, namely: K_x , K_p , G and U (see Figure 3), to have a visualization of such dependencies.

To summarize:

Increasing the size of the user base of a FLOSS program (U), has no direct effect on the fraction of inter-release time (T) exploited bugs live. However, the total exploitation time (τ_x), is reduced with increasing U , because T is reduced.

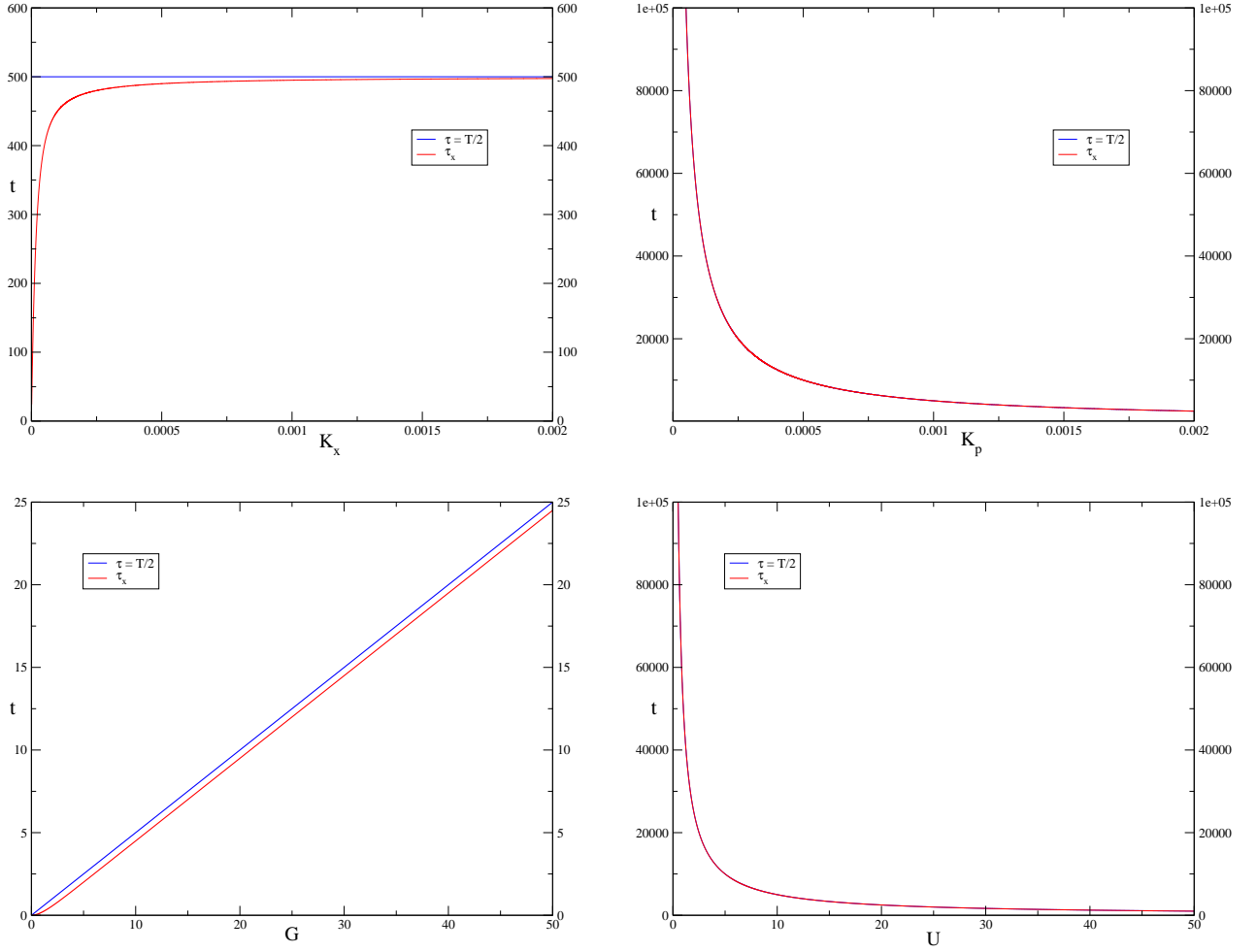


Figure 3: Evolution of average bug life time (τ) and exploited bug lifetime (τ_x) with K_x (top left), K_p (top right), G (bottom left) and U (bottom right), with remaining parameters constant: $K_x = 0.01$, $K_p = 0.01$, $G = 1000$ and $U = 100$. Recall that in the K_p and U plots τ and τ_x are indistinguishable.

3 Conclusions

The “Increasing popularity = Increasing bugginess” motto is a *non sequitur*. According to the simple model outlined here, the broader the user community of program \mathcal{P} , the faster bugs will be patched, even admitting that an increase in user base brings an equal increase in the number of crackers committed to doom \mathcal{P} . Even for crackers that are more effective in their cracking work than the *bona fide* users in their patching work ($K_x \gg \gg K_p$), increasing the community size **does** reduce how long the bugs stay unpatched and **also** how long the exploited bugs stay unpatched. No matter how clumsy the users, and how rapacious the crackers, the free model (whereby the users are granted access to the code, and thus empowered to contribute to \mathcal{P}) ensures that popularization **is** positive, for both \mathcal{P} and the community itself.

Compare that with a closed model, in which an increased user base may boost the number or crackers attacking a program, but certainly adds little, if anything, to the code patching and correcting speed. It is actually proprietary software that should **fear** popularization. It is easy to see that when a particular proprietary software piece grows over a certain “critical mass” of

users, the crackers could potentially disrupt its evolution (say, $\tau_x = T$, $F_x = 1$), because G , P and thus T , are kept constant (depend only on the sellers of the code).

4 Copyright

This work is licensed under the Creative Commons Attribution-Noncommercial-Share Alike 2.5 License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-sa/2.5/> or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.