

¿Popularización de Software Libre acentúa explotación de bugs?

Iñaki Silanes

Euskal Herriko Unibertsitatea (UPV/EHU)
Donostia, España

November 23, 2006

1 Introducción

Los excépticos del movimiento FLOSS suelen decir que el Software Libre tiene, en general, menos bugs explotados que el software privativo, solamente porque es menos popular que este. Argumentan que, como el FLOSS tiene menos usuarios, los crackers estarán menos interesados en malgastar su tiempo intentando explotar los bugs que pudiera tener. La mayor base de usuarios del software privativo daría además, según ellos, una mayor publicidad a sus bugs, y sus modos de explotación se difundirían más rápido. El corolario a esta teoría sería que la popularización de aplicaciones FLOSS (p.e. Firefox), llevaría a un incremento en el número de bugs descubiertos y explotados, llegando eventualmente a un estado similar al del software privativo actual (p.e. “*Cuando Firefox sea tan popular como Internet Explorer, tendrá tantos bugs como Internet Explorer.*”).

El objetivo de este artículo es demostrar matemáticamente la total insensatez de tal teoría. Específicamente, argumentaré que un aumento en el tamaño de la comunidad de un proyecto FLOSS lo mejora de al menos 3 formas:

1. Desarrollo acelerado
2. Menor vida media de bugs abiertos
3. Menor vida media de bugs explotados

2 Proposiciones y desarrollo

Imaginemos que tenemos un proyecto FLOSS \mathcal{P} . Postulemos que los desarrolladores publicarán nuevas versiones con un período \mathbf{T} , cada tal versión incorporando \mathbf{G} nuevos bugs. Consideremos también, por simplicidad, que cada nueva versión es liberada cuando todos los bugs de la anterior han sido parcheados.

Asumiré también que habrá una velocidad de parcheo, \mathbf{P} , que depende del tamaño de la comunidad, \mathbf{U} , usando, testeando y contribuyendo a \mathcal{P} . Si asumimos que esta dependencia sea una *proporcionalidad*, obtendremos la Ecuación 1, donde K_p es una constante que puede entenderse como la “eficiencia de parcheo” de la comunidad.

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

Dado que P es la velocidad con la que los bugs abiertos (β) son parcheados, la dependencia temporal de β puede ser calculada de la Ecuación 1, y tal se hace en la Ecuación 2.

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

El período de tiempo entre versiones de \mathcal{P} (T), puede derivarse de la simple Ecuación 2. Resolviendo para $\beta = 0$ obtenemos la Ecuación 3.

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

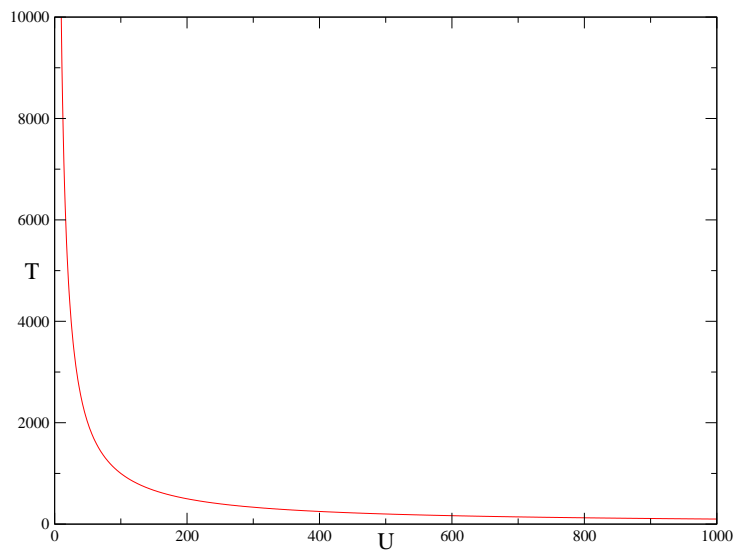


Figure 1: Tiempo entre versiones (T) frente a la cantidad de usuarios en la comunidad (U), según la Ecuación 3, para $K_p = 0.01$ y $G = 1000$, constantes.

La interpretación de la Ecuación 3 es inmediata: cuantos más bugs en una versión (G), más tardará en salir la siguiente. Por el contrario, cuantos más usuarios (U), y/o más rápido parchee de media cada uno (K_p), más rápido saldrán las versiones, o, en otras palabras:

**Aumentar el tamaño de la base de usuarios de un programa FLOSS
acelera su desarrollo.**

2.1 Vida media de los bugs

En el período entre t y $t+dt$, $(-d\beta/dt)dt$ bugs serán parcheados (velocidad de parcheo por intervalo de tiempo). Tomemos en cuenta que $d\beta/dt$ es negativo. La edad de estos bugs (cuanto tiempo llevan abiertos) sería t . La vida media de los bugs sería, por tanto, la suma de las edades de todos los bugs, multiplicada por el número de bugs de *esa* edad, y dividido por el número total de bugs. Si llamamos τ a dicha media, podemos hacer el desarrollo de la Ecuación 4.

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

Lo que nos dice la Ecuación 4 es que la vida media de los bugs será siempre la mitad del período entre versiones (T). Ver Figura 1 para la evolución de T con U . Esto significa que τ , al igual que T , es inversamente proporcional al tamaño de la comunidad, o, en otras palabras:

Incrementar el tamaño de la base de usuarios de un programa FLOSS reduce el tiempo que los bugs permanecen abiertos.

2.2 Fracción de bugs que son explotados antes de ser parcheados

Si quisieramos calcular cuántos bugs son de hecho explotados durante su vida, tendríamos que definir una “velocidad de explotación de bugs”, que propongo sea proporcional al tamaño de la comunidad (U). La razón de ello es que cuanta más gente use \mathcal{P} , más crackers habrá entre ellos, y también más “interesante” será para el cracker explotar sus bugs. Esta velocidad sería también proporcional al número de bugs abiertos *y* todavía no explotados, β_{ou} . Si tal fuera el caso, obtendríamos la Ecuación 5.

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

Para resolver la Ecuación 5 uno tiene que tomar en cuenta las Ecuaciones 6 – 8, donde β_{ox} y β_{px} corresponden a la cantidad de bugs explotados que están, respectivamente, todavía abiertos o ya parcheados.

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

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

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

Las Ecuaciones 6 – 8 ayudan a calcular la evolución temporal tanto de β_{ou} como de β_{ox} . Para β_{px} tenemos la Ecuación 9.

$$\frac{d\beta_{px}}{dt} = \overbrace{\left(-\frac{d\beta}{dt}\right)}^{\text{velocidad de parcheo}} \underbrace{\left(\frac{\beta_{ox}}{\beta}\right)}_{\text{fracción explotada}} = K_p U \frac{\beta_{ox}}{\beta} \quad (9)$$

Necesitamos β_{ox} de antemano para resolver la Ecuación 9. Para obtener dicho valor, haremos el desarrollo de la Ecuación 10.

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

Uniendo las Ecuaciones 2, 8 y 10, obtenemos una expresión para β_{ox} , concretamente la Ecuación 11.

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

Podemos ahora desarrollar la Ecuación 9, incluyendo la fórmula en la Ecuación 11, lo cual produce la Ecuación 12.

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

De las Ecuaciones 7, 11 y 12, podemos calcular el total de bugs explotados (abiertos y parcheados) en cualquier momento (β_x), y dividiendo por el total de bugs generados (G), obtener

la fracción de bugs que acaban siendo explotados en algún momento. Esta magnitud se da en la Ecuación 14, donde γ está definida en la Ecuación 13.

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

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

Desearíamos saber la fracción total de bugs que acaban siendo explotados para cuando todos los bugs han sido parcheados, esto es $F_x = \beta_x(T)/G$. Para ello usaré las Ecuaciones 3 y 14, obteniendo la Ecuación 15.

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

De la Ecuación 15 se deduce que la fracción de los G bugs iniciales que acaba siendo explotada es **independiente de U** , y únicamente depende de γ , esto es, de la relación $K_p/K_x G$. Esto tiene sentido, si tenemos en cuenta que he propuesto que cuanto más amplia sea la comunidad de usuarios de \mathcal{P} , más rápido serán parcheados los bugs (Ecuación 1), pero **también** más rápido serán explotados (Ecuación 5).

Se puede hacer un chequeo de la Ecuación 15, probando los límites superior e inferior de γ , esto es, cuando la cantidad de bugs iniciales (G) y/o la efectividad de los crackers para hacer su trabajo sucio (K_x), es mucho mayor ($\gamma \rightarrow 0$) y mucho menor ($\gamma \rightarrow \infty$) que la efectividad de la comunidad para parchear los bugs (K_p). Este chequeo se presenta en la Ecuación 16, y confirma la idea intuitiva de que para una comunidad infinitamente eficiente en el parcheo, todos los bugs serían parcheados **antes** de ser explotados. En el caso opuesto, todos los bugs acabarán siendo explotados tarde o temprano, en sus largas vidas. La dependencia de F_x con γ se representa en la Figura 2.

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

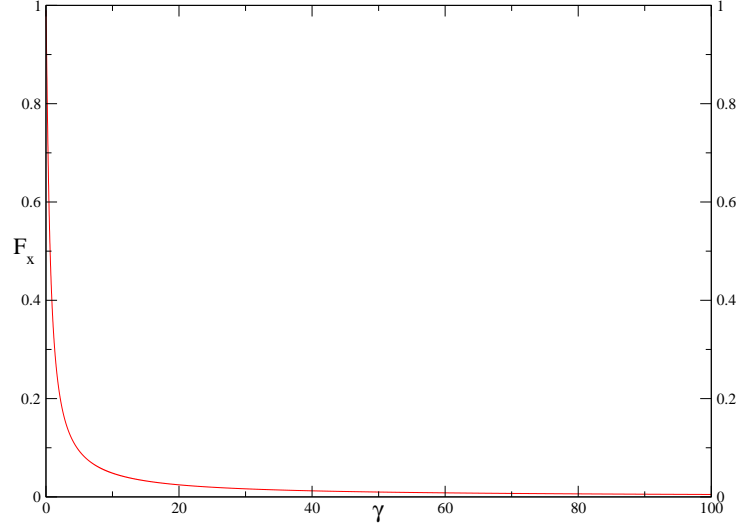


Figure 2: Evolución de F_x (ver Ecuación 15) con γ (ver Ecuación 13).

2.3 Vida media de bugs explotados

Me preocupa ahora cuánto tiempo permanecen abiertos los bugs que han sido explotados. Empezaré asumiendo que los bugs explotados son parcheados a la misma velocidad que el resto. Hay que tener en cuenta que esto supone sobreestimar el tiempo que los bugs permanecen explotados, ya que normalmente los bugs explotados son más diligentemente atendidos que aquellos que se conocen, pero no han sido todavía explotados.

Escribo a continuación las magnitudes que necesitaré para proseguir la discusión:

1. Bugs explotados entre t y $t + dt$ (Ecuación 5):

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

2. Fracción del total de bugs abiertos que son explotados entre t y $t + dt$:

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

3. Fracción del total de bugs abiertos en t' que corresponde a los que fueron explotados en t :

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

4. Total de bugs explotados que fueron explotados hace $\Delta t = t' - t$, en t' :

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

5. Número de bugs explotados parcheados en t' (Ecuaciones 1 y 8):

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

6. Fracción del total de bugs abiertos que representan los explotados parcheados en t' :

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

7. Bugs de edad de explotación Δt parcheados en t' :

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

Podemos integrar el producto de $p(\Delta t)$ arriba por el edad de explotación (Δt), y dividir por el total de bugs explotados en algún momento ($\beta_{px}(T)$), para obtener la media de tiempo de explotación de los bugs que han sido efectivamente explotados. El correspondiente desarrollo se da en al Ecuación 17, donde τ y F_x son los mismos de las Ecuaciones 4 y 15, respectivamente.

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

Como puede verse, la media de tiempo que los bugs explotados permanecen sin parchear es proporcional al tiempo entre versiones de \mathcal{P} (T). La constante de proporcionalidad (F_x) depende solamente de la previamente definida γ , y es por tanto **independiente** de U . Para γ s pequeños ($K_p \ll K_x G$), F_x se aproxima a la unidad, de forma que $\tau_x \sim \tau = T/2$. Por el contrario, cuanto mayor sea γ (K_p aumenta con respecto a $K_x G$), menor será τ , o, en otras palabras, menor el tiempo de explotación, incluso respecto a T . Puntualicemos, sin embargo, que el propio T es también afectado por U , como se ve en la Ecuación 3.

Podemos representar τ_x frente a todos los parámetros de los que depende, esto es: K_x , K_p , G y U (ver Figura 3), para visualizar dichas dependencias.

En resumen:

Aumentar el tamaño de la base de usuarios (U) de un programa FLOSS no tiene un efecto directo sobre la fracción de tiempo entre versiones (T) que los bugs explotados sobreviven. Sin embargo, el tiempo de explotación absoluto (τ_x) sí se reduce, ya que T se reduce al aumentar U .

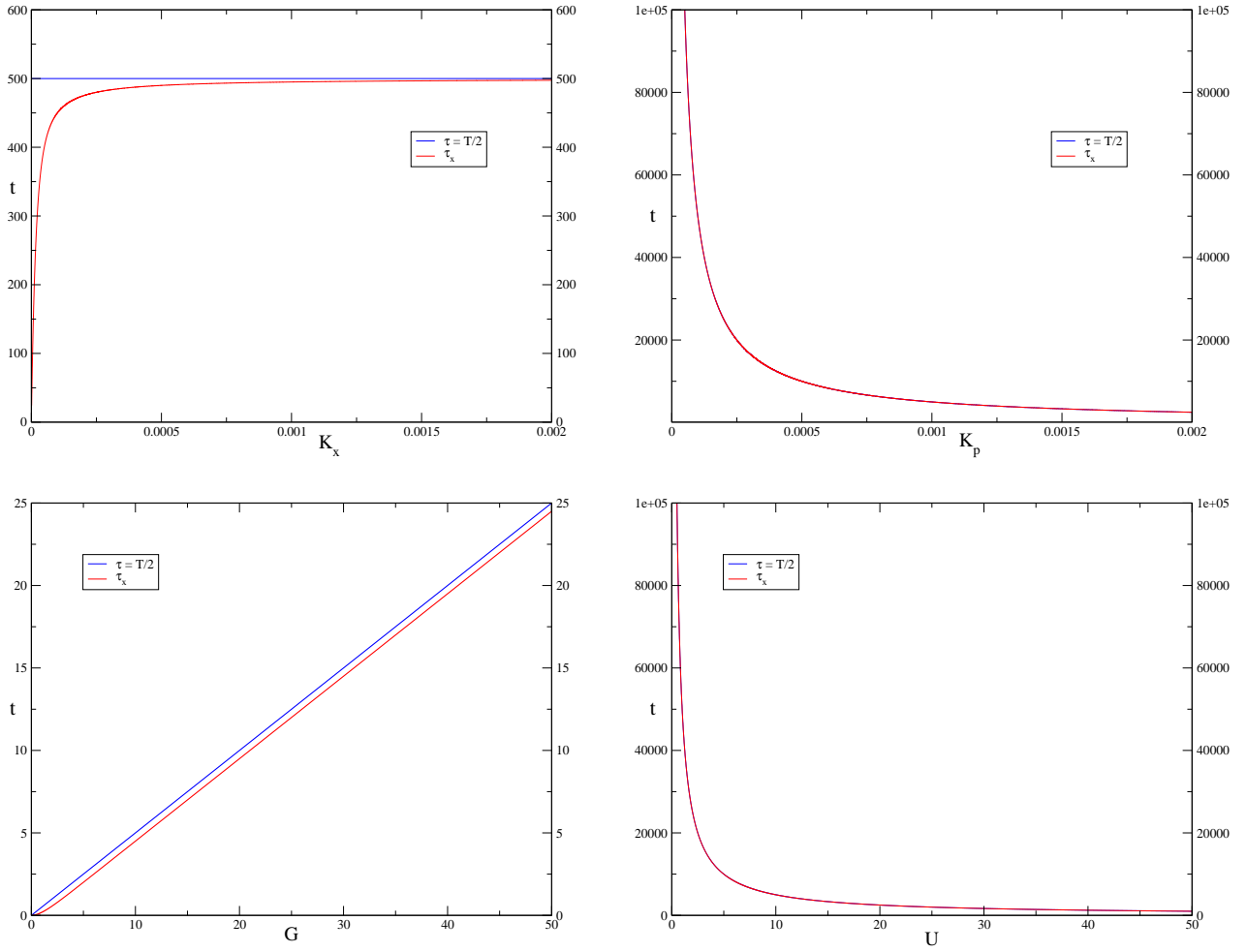


Figure 3: Evolución de la vida media de un bug (τ) y el tiempo de explotación medio de un bug (τ_x) respecto a K_x (arriba izquierda), K_p (arriba derecha), G (abajo izquierda) y U (abajo derecha), siendo el resto de parámetros constantes: $K_x = 0.01$, $K_p = 0.01$, $G = 1000$ y $U = 100$. Resaltemos que en las gráficas de K_p y U , τ y τ_x son coincidentes.

3 Conclusiones

El eslogan “Más popularidad = más bugs” es un *non sequitur*. De acuerdo con el simple modelo que bosquejo aquí, cuanto más amplia sea la comunidad de usuarios de \mathcal{P} , más rápido serán parcheados los bugs, incluso admitiendo que a más usuarios, más crackers entre ellos, dispuestos a acabar con \mathcal{P} . Incluso para crackers que sean más efectivos en su desempeño que los usuarios de buena fe en su trabajo de parcheo ($K_x \gg \gg K_p$), aumentar el tamaño de la comunidad **reduce** el tiempo que los bugs permanecen abiertos y **también** cuánto tiempo tardan en parchearse los bugs ya explotados. No importa cuán torpes sean los usuarios, y cuán rapaces los crackers, el modelo libre (por medio del cual se da acceso al código a los usuarios, dándoles así poder para contribuir a \mathcal{P}) asegura que la popularización **es** positiva, tanto para \mathcal{P} como para la propia comunidad.

Comparemos esto con un modelo cerrado, en el que una base de usuarios mayor puede incrementar el número de crackers atacando a un programa, pero ciertamente añade poco o nada a la velocidad con que el código es parcheado y corregido. Es de hecho el software privativo el que

debe **temer** su popularización. Es fácil de ver que cuando una pieza determinada de software privativo alcanza una cierta “masa crítica” de usuarios, los crackers pueden potencialmente desmoronar su evolución (digamos, haciendo $\tau_x = T$, $F_x = 1$), porque (a diferencia del FLOSS), G, P, y por lo tanto T, son constantes (ya que dependen únicamente de los vendedores del software).

4 Copyright

Esta obra est bajo una licencia Reconocimiento-No comercial-Compartir bajo la misma licencia 2.5 Espaa de Creative Commons. Para ver una copia de esta licencia, visite:

<http://creativecommons.org/licenses/by-nc-sa/2.5/es/>

o envíe una carta a Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.