EADS INNOVATION WORKS

La sécurité informatique est-elle dépendante du logiciel libre ?

Louis Granboulan
11 octobre 2012

Open World Forum – session « Security and Free Software »

EADS

Réponse :

Oui

Mais on peut détailler…

Qu'est-ce que le logiciel libre apporte à la sécurité ?

(sécurité informatique, en particulier)

En quoi le fait d'être libre a favorisé cet apport ?

(qu'est-ce qui est différenciant ?)

Le logiciel propriétaire a-t-il évolué à cause du libre ?

Un peu de bibliographie

La question habituellement posée est : le logiciel libre est-il moins vulnérable que le logiciel propriétaire ?

Le livre *Secure Programming for Linux and Unix HOWTO* par David A. Wheeler contient un chapitre résumant les discussions sur le sujet au début des années 2000

http://www.dwheeler.com/secure-programs/Secure-Programs-HOWTO/open-source-security.html

Et un guide a été publié par le SANS en 2003

http://www.sans.org/reading_room/whitepapers/awareness/security-concerns-open-source-software-enterprise-requirements_1305

Le débat en est à peu près au même point dix ans plus tard

http://manoharbhattarai.wordpress.com/2010/07/24/why-free-software-is-more-secure/

http://www.techrepublic.com/blog/security/key-open-source-security-benefits/4941

Mais il continue à être vivant…

Un workshop *Open source et sécurité* a eu lieu le 13 septembre dernier dans le cadre des Labs Hadopi

Quelques exemples de logiciels libres essentiels pour la sécurité

Services de base : apache, sendmail, squid, spamassassin, vsftpd, openvpn, …

Outils de base : GNU, Linux, mysql, python, …

Développement logiciel : cvs, trac, sonar, findbugs, …

Logiciels de sécurité informatique : gnupg, openssl, truecrypt, metasploit, john, snort, scapy, ossec, nufw, …

Et aussi des logiciels pour la sécurité globale, par exemple d'analyse de données : tesseract, libface, CMU sphinx, …

Question directrice

Le fait d'être un logiciel libre est-il différenciant ?

1. Accès au code source

Possibilité d'audit de sécurité par l'utilisateur

La publication des algorithmes devient la norme en cryptographie, pour des raisons de sécurité ; pourquoi pas la même chose en matière de logiciel ? Question par exemple posée par B. Schneier en 1999…

Mais en réalité, c'est le processus de développement et surtout d'évaluation qui est important

Il doit y avoir une évaluation de la sécurité du logiciel (erreurs de programmation, ou d'architecture) et de la sécurité du logiciel dans son environnement (validation que l'utilisation est conforme aux hypothèses de l'évaluation de sécurité)

Se pose aussi la question de quel code source évaluer… Exemple de la faille openssl sur Debian en 2008, ou bien des versions Apple de logiciels open source

En pratique

Les logiciels open source très utilisés sont continument évalués par la communauté des experts en sécurité, les logiciels peu utilisés ne sont pas évalués… et la qualité de l'éventuelle évaluation n'est pas mesurée

1. Accès au code source

Possibilité de patch des failles

Car un logiciel libre n'est pas seulement un logiciel de source public, il peut être modifié par n'importe quel utilisateur

Avantage du logiciel libre

Les patchs de sécurité peuvent être publiés plus vite (e.g. l'étude Firefox vs. IE faite par Secunia en 2008, montrant un délai moyen de 43 jours pour Mozilla et 110 jours pour Microsoft)

Celui qui découvre une faille peut la corriger lui-même

Inconvénients du logiciel libre

Le processus de validation des patchs est en général incomplet

Non-prédictibilité de la diffusion des patchs, donc déploiement compliqué

1. Accès au code source

Possibilité de modification du logiciel

C’est souvent la raison du choix d’un logiciel libre

Extensions

Le rajout de fonctions peut améliorer la sécurité (patch grsecurity!)

Mais en général cela l’empire…

Simplifications

Il n’est pas rare, pour des raisons de sécurité, d’enlever les parties du logiciel dont on ne se servira pas

C’est par exemple un usage courant chez EADS pour les logiciels assurant des fonctions critiques

1. Accès au code source

Impact sur la sécurité des logiciels propriétaires

Possibilité d'audit de sécurité par l'utilisateur

- L'audit de sécurité des logiciels propriétaires, par leur client ou un tiers de confiance, n'est pas rentré dans l'usage
- Certains clients commencent à le demander systématiquement : systèmes critiques, agences gouvernementales, …

Possibilité de patch des failles

- Lorsqu'un concurrent open source existe, la comparaison pousse l'éditeur commercial à améliorer sa réactivité

Possibilité de modification du logiciel

- Des systèmes de plugins apparaissent de plus en plus fréquemment, souvent au détriment de la sécurité

2. Facilité de déploiement

Pas de processus d'achat

L'utilisation d'un logiciel libre peut se faire de l'initiative de l'administrateur du système

- Plus grande réactivité, les processus d'achats étant parfois très longs
- Pas d'obstacle à avoir toujours la version la plus récente

Les frais d'acquisition sont cachés

- Ils se résument à la montée en compétence des administrateurs et utilisateurs
- Il est plus facile de proposer des alternatives redondantes (e.g. plusieurs browsers)
- Le déploiement de maquettes de test est facilité

Le libre a la faveur des utilisateurs experts en informatique

- Beaucoup utilisé pour le déploiement de logiciels de sécurité informatique (firewalls, supervision, tests de sécurité)
- Beaucoup utilisé dans le milieu de la recherche

2. Facilité de déploiement

Inclusion de bibliothèques ou de modules libres

Accélère le développement de logiciels

- La variété de l'écosystème du libre permet d'y trouver tout ce qui n'est pas la fonction innovante qu'on veut inclure dans notre logiciel
- Mais il faut savoir choisir (gnome vs. Qt, openssl vs. GnuTLS ou NSS, etc.)

Évite de mal réinventer la roue

- Par exemple en matière d'implémentations cryptographiques, il est rare que les développeurs aient les compétences pour faire mieux que ce qui est publiquement disponible
- Pareil pour la gestion de la mémoire ou des chaînes de caractères
- Mais vaut-il mieux développer en C en utilisant des bibliothèques tierces, ou bien utiliser un langage de programmation dont le toolkit fournit ces outils?

2. Facilité de déploiement

Impact sur la sécurité du logiciel propriétaire

Pas de processus d'achat

- Les vendeurs proposent en général des versions d'évaluation gratuitement
- Pour un intégrateur comme EADS, c'est essentiel !

Inclusion de bibliothèques ou de modules libres

- Les logiciels propriétaires sont nombreux à inclure du libre (non viral, en général)
- Le processus de mise à jour de ces logiciels doit donc tenir compte des mises à jour de ces modules libres… exemple d'Apple qui rajoute un délai de validation des patchs de sécurité
- Un industriel développant du logiciel pourra être tenté de produire du logiciel libre pour pouvoir inclure du code GPL ; donc son processus de développement s'adapte à la possibilité que le code source devienne public

3. Communautés

Communauté de développeurs

Lisibilité du code

- Nécessaire pour que les développeurs travaillent efficacement
- Impact positif sur la sécurité
- En réalité, la lisibilité n'est pas souvent bonne…

Acceptation des patchs

- En théorie, la revue des patchs par des tiers est un facteur de qualité
- En pratique, l'acceptation se décide plutôt sur des critères humains

Packaging

- En théorie, la portabilité améliore la sécurité
- En pratique, on voit souvent des tests ad hoc pour détecter l'environnement…

3. Communautés

Communauté d'utilisateurs

Retour d'expérience

- Les développeurs mettent en place des listes de discussion, des outils de remontée de bugs, où il est possible de fournir des éléments d'analyse du bug et de proposer des correctifs

Demande d'ajout de fonctionnalités

- L'impact sur la sécurité peut être important, si le processus de développement n'impose pas une analyse de sécurité pour chaque ajout de fonction

L'efficacité de la communauté d'utilisateurs est améliorée quand le sentiment communautaire est fort

- Apple réussit à faire cela même sur ses logiciels propriétaires !

3. Communautés

Impact sur la sécurité du logiciel propriétaire

Communauté de développeurs

Il est courant de voir un logiciel propriétaire devenir libre pour bénéficier d'une communauté de développeurs (Netscape/Mozilla et StarOffice/OpenOffice étant les plus célèbres)

Communauté d'utilisateurs

Les logiciels propriétaires assument en général l'existence de bugs et prévoient un moyen de remonter les informations. Soit c'est totalement automatique (e.g. rapports de plantage) soit ça s'inspire du libre (mêmes outils) soit c'est artisanal (adresse de contact)

L'ajout de fonctionnalités quant à lui est strictement contrôlé

Conclusion

Le libre est important

- Parce qu'il propose des pratiques différentes
- Parce qu'il propose des alternatives

Mais

- Le logiciel propriétaire peut avoir les mêmes qualités que le libre, il suffit qu'il s'en inspire