
Sécurité et logiciel libre

Cédric Blancher <blancher@cartel-securite.fr>

—

Rencontres Mondiales du Logiciel Libre
10 juillet 2003

Lorsqu'un logiciel est LIBRE, vous pouvez :

- ▶ l'utiliser librement (L0)
- ▶ accéder à son code source et l'étudier (L1)
- ▶ le redistribuer librement (L2)
- ▶ le modifier et distribuer vos modification (L3)



Un système de sécurité est un équilibre éclairé entre :

- ▶ le risque associé au SI qu'on protège (C1)
- ▶ la coût des protection mises en place (C2)
- ▶ la pertinence des solutions déployées (C3)



- La problématique de confiance
 - ▶ Pourquoi fait-on de la sécurité ?
 - ▶ Peut-on avoir confiance en ses outils ?

- La problématique d'indépendance
 - ▶ Peut-on être en sécurité si on est dépendant ?
 - ▶ La dépendance nuit-elle à notre sécurité ?

- Discussion ouverte
 - ▶ Validité des argumentations classiques
 - ▶ Dégager des vrais axes de réflexion et de promotion



- La problématique de confiance

- ▶ Pourquoi fait-on de la sécurité ?
- ▶ Peut-on avoir confiance en ses outils ?

- La problématique d'indépendance

- ▶ Peut-on être en sécurité si on est dépendant ?
- ▶ La dépendance nuit-elle à notre sécurité ?

- Discussion ouverte

- ▶ Validité des argumentations classiques
- ▶ Dégager des vrais axes de réflexion et de promotion



La problématique de confiance :

- ▶ Le besoin de sécurité découle du manque de confiance
 - Je me connecte à un monde inconnu
 - Je ne connais pas ce monde
 - Je ne fais pas confiance
- ➔ On fait de la sécurité pour retrouver de la confiance



Les outils mis en jeu :

- ▶ Équipements
 - ▶ Systèmes d'exploitation
 - ▶ Logiciels mis en œuvre
 - ▶ Outils de sécurité
- ➔ La sécurité implique la confiance en ces outils



Qu'est-ce qu'un logiciel de confiance ?

Les trois règles d'or de l'assurance qualité en ingénierie logicielle :

- ▶ Je dis ce que je fais
- ▶ Je fais ce que je dis
- ▶ Je prouve que je le fais



L'outil est-il adapté (C3)

- ▶ Vous savez ce que vous allez utiliser
- ▶ Ce que vous utilisez répond bien aux spécifications
- ➔ Description exacte du produit



Il prouve qu'il le fait (C3)

- ▶ Vous disposer d'un moyen de vérifier ce qu'il fait



Logiciel libre :

- ▶ Sources disponibles (L1)
 - ↳ Nous pouvons vérifier ce que fait le programme
 - ↳ Nous pouvons vérifier comment il le fait
 - ↳ Nous pouvons compiler le programme depuis les sources



Logiciel propriétaire :

- ▶ Nous ne disposons pas des sources en général
 - ➔ Nous ne pouvons pas vérifier le code
- ▶ Lorsque nous avons la chance de les voir, nous ne pouvons pas les utiliser
 - ➔ Nous ne pouvons pas être sur que les programmes que l'on utilise proviennent des sources que l'on nous a montrés



- La problématique de confiance

- ▶ Pourquoi fait-on de la sécurité ?
- ▶ Peut-on avoir confiance en ses outils ?

- La problématique d'indépendance

- ▶ Peut-on être en sécurité si on est dépendant ?
- ▶ La dépendance nuit-elle à notre sécurité ?

- Discussion ouverte

- ▶ Validité des argumentations classiques
- ▶ Dégager des vrais axes de réflexion et de promotion



Indépendance vis-à-vis de l'éditeur :

- ▶ Patches de sécurité (L1+L3)
- ▶ Ajout de fonctionnalités (L1+L3)
- ▶ Suivi du logiciel (L1+L3)
- ↳ Pérennité de la solution (C2)



Indépendance vis-à-vis du fournisseur :

- ▶ Disponibilité (L2)
- ▶ Maintenance (L1+L3)
- ▶ Déploiement (L0)
- ▶ Services (L1+L2+L3)
- ➔ Pérennité de la solution (C2)



Pas de licence contraignante :

- ▶ Conditions d'utilisation libres (L0)
- ▶ Pas de limite d'utilisation (L0)
- ▶ Pas de contrainte d'obtention (L2)
- ▶ Possibilité d'adapter le logiciel (L1+L3)
- ➔ Universalité de la solution (C2+C3)



Ce qu'il éviter

- ▶ Dépendance technologique (<>C3)
- ▶ Dépendance économique (<>C2)



Dépendance technologique

- ▶ Incapacité d'évaluer la technologie
 - ▶ Incapacité de qualifier sa sécurité
 - ▶ Incapacité de valider une migration
- ➡ Cercle vicieux



Dépendance économique

- ▶ Problématique de l'Intelligence Économique
- ▶ Problématique des intérêts divergents
- ▶ Incapacité de maîtriser sa sécurité en terme de coût



- La problématique de confiance
 - ▶ Pourquoi fait-on de la sécurité ?
 - ▶ Peut-on avoir confiance en ses outils ?
 - La problématique d'indépendance
 - ▶ Peut-on être en sécurité si on est dépendant ?
 - ▶ La dépendance nuit-elle à notre sécurité ?
- Discussion ouverte
 - ▶ Validité des argumentations classiques
 - ▶ Dégager des vrais axes de réflexion et de promotion



- ▶ Windows 9x/NT/2000 :
195 failles
- ▶ Solaris : 98 failles
- ▶ RedHat Linux : 96 failles
- ▶ Debian GNU/Linux :
54 failles
- ▶ OpenBSD : 14 failles

« more than 50 percent of all security advisories that CERT issued in the first 10 months of 2002 were for Linux and other open-source software solutions »



ÇA N'A AUCUN SENS !



- ▶ Exceptionnelles
- ▶ Découvertes
- ▶ On peut les enlever

- ▶ Vraies backdoors
- ▶ Spywares
- ▶ Prises de contrôle à distance pour faciliter la maintenance par l'éditeur, mais cachée au client
- ▶ Key excrows
 - ➔ Mots de passe constructeurs (BIOS, ...)
 - ➔ Windows encryption keys
 - ➔ Lotus Notes et le gouvernement suédois (clefs de 64 bits dont 24 connus des États-Unis)



- ▶ Exceptionnel, sous la forme d'obfuscation
- ▶ Cryptographie d'amateur (e-book, ...)
 - ▶ Faux sentiment de sécurité, argument marketing
 - ↳ ex: GSM
 - ↳ ex: exploit Cisco sans les sources



- ▶ Full disclosure
 - ▶ Il suffit de regarder le patch
 - ▶ Possibilité de juger par soi-même de la criticité d'une faille, en prenant en compte son contexte.
- ▶ Problème plus marketing que technique (cf. sécurité par l'obcurité)
 - ▶ Problème d'image (avertir ou garder secret)
 - ↳ ex: banques, retraits fantômes et algorithmes faibles
 - ▶ Criticité évaluée en fonction d'une cible de marché, et actions prises en conséquence



- ▶ Quasi-systématique
- ▶ Rapide (en moyenne quelques jours, parfois dans la demi-heure, parfois le patch est livré par le découvreur)
- ▶ Dépendante du bon vouloir de l'éditeur
- ▶ Produits plus maintenus : pas de solution (ex: NT)
- ▶ Failles considérées comme non critique : pas de solution



- ▶ On peut corriger soi-même
 - ▶ On peut payer quelqu'un pour le faire
 - ▶ La communauté peut reprendre un projet
 - ↳ Linux 2.0
 - ↳ Debian Potato
 - ▶ On peut ajouter des fonctionnalités
- ▶ Dépendant du bon vouloir de l'éditeur
 - ▶ ex: NT n'est plus maintenu, et personne ne peut plus rien y faire.



- ▶ Garantie des libertés d'utilisation et de distribution
 - ▶ Un expert peut utiliser la pleine mesure des outils
- ▶ Problèmes de licences
 - ▶ Utilisation des outils limitée à un budget
 - ↳ Une partie des postes sans antivirus
 - ↳ Une partie des postes sans firewall
 - ↳ Pas de mises à jour car trop chères
 - ↳ ...
 - ▶ Un expert ne pourra pas dépasser les limites arbitraires des licences

