Synthèse du workshop open-source et sécurité

 [1]
Document de travail
Tag(s) :
logiciel libre [2] open-source [3] sécurité [4]
Par Tris [1], le 21-09-2012

Jeudi 13 septembre s'est tenu le workshop open-source et sécurité à la Mutinerie, animé par Bruno Spiquel.

Le document présenté ici est la synthèse des débats qui s'y sont tenus, fruit d'un travail collectif des participants. Nous tenons à remercier ceux qui sont venus et qui ont échangé autour de cette thématique.

Ce texte est au format Wiki, il peut donc faire l'objet de modifications.

1 | Postulat de base

Ce passage est un rappel de ce qui avait été énoncé ici [5]

Lorsque l'intelligence logicielle était captive du matériel, la sécurité informatique était une préoccupation très secondaire voire inexistante.

La séparation du hardware (au sens de matériel) et du software a donnée naissance à la possibilité d'exploiter les failles logicielles. L'open-source a beaucoup facilité cet accès, rendant les failles analysables et exploitables par à peu près n'importe qui.

Pour autant, il semble peu réaliste de revenir en arrière pour enfermer à nouveau le software dans du hardware dans le but de le protéger En fait, l?open-source permet

finalement de curer le « mal » qu'il a lui-même contribué à créer, pour peu que tous se donnent la peine d'entretenir, à leur niveau, leur univers logiciel.

2 | Les réactions en ligne au postulat

Suite à la publication du thème du workshop précédemment énoncé, des internautes présents ont réagi et ont émis certaines réserves.

Sur la question de la sécurité, il a été souligné qu?il n?était pas forcément nécessaire de procéder à la rétro-ingénierie pour connaître les sources d?un logiciel C??est surtout l?intelligence collective de la communauté de l?open-source qui permet de corriger des failles de sécurité. Par ailleurs, de ce point de vue, le logiciel et le matériel sont égaux, puisque le matériel peut aussi être open-source.

Les internautes se sont également interrogés sur la démocratisation de la sécurité informatique et de l?open-source. Pour certains, c?est le nombre d?utilisateurs qui impacte la sécurité informatique. Pour d?autres, c?est l?influence en entreprise, qui peut amener une personne à reproduire dans un cadre privé, les habitudes acquises au travail, notamment en matière de sécurité et d?open-source.

Mais la question de la rétro-ingénierie soulève des subtilités juridiques difficiles à appréhender pour celui qui n?est pas juriste. Selon le support, c?est le droit de la propriété intellectuelle qui s?applique ou le droit de la propriété industrielle, avec des aspects de droit international privé.

3 | La sécurité

De façon générale, la notion de sécurité s?appuie sur un contrat social. Si on souhaite faire une analogie, on peut prendre l?exemple de la conduite. N?importe qui peut, sur la route, donner un coup de volant pour provoquer une collision frontale, mais il n?y a que très peu de personnes ? sur l?ensemble des conducteurs ? qui font cela sciemment.

Aux débuts de l?informatique, la notion de sécurité reposait également sur cette notion de contrat social, dans la mesure où les réseaux et l?informatique étaient principalement exploités par les chercheurs.

4 | Open V. Closed source : état des lieux

Il convient de différencier l?open-source, du libre, du closed source.

Selon Wikipedia, la désignation open source s'applique aux logiciels dont la licence respecte des critères précisément établis par l'Open Source Initiative, c'est-à-dire la possibilité de libre redistribution, d'accès au code source et aux travaux dérivés.

Toujours selon Wikipedia, un logiciel libre est un logiciel dont l'utilisation, l'étude, la modification et la duplication en vue de sa diffusion sont permises, techniquement et légalement. Ceci afin de garantir certaines libertés induites, dont le contrôle du programme par l'utilisateur et la possibilité de partage entre individus

Le code source constitue l'origine de tout programme. Avant de fonctionner un logiciel doit être compilé, le code source disparaît alors et laisse place à un programme interprétable par la machine qui n'est, *a priori*, pas modifiable.

Le closed source correspond au fait de fournir le programme utilisable sans donner les sources. L'open-source permet de modifier le logiciel comme on le souhaite, chacun étant en mesure de compiler le logiciel à partir de la source et obtenir la version utilisable par l'ordinateur, incluant ses propres modifications.

Il a également été fait mention du logiciel privé, qui est un outil développé pour un utilisateur spécifique [6].

La question économique de l?open-source soulève souvent des interrogations car la plupart des logiciels open-source sont gratuits. Le modèle économique de l?open-source repose, non pas sur la vente de licences comme pour le logiciel propriétaire, mais sur le support et le développement particulier, personnalisé, adapté.

L?avantage de l?open-source réside également dans la transparence. L?utilisateur peut voir ce qui se passe dans le soft avant de l'activer, bien que cette possibilité ne soit réellement utilisée que par une frange très réduite de la population. La simple existence de la possibilité est rassurante.

5 | Historique du libre et de l?open-source

Il est intéressant de revenir historiquement sur les grandes dates du libre et de l?open-source. Les dates indiquées ci-dessous sont à inscrire dans une perspective plus large [7].

· 1946 : les premiers calculateurs sont crées

· 1952 : premier ordinateur conçu par IBM

· 1958 : naissance d?ARPA

· 1962 : mise en place de l?IPTO, programme de recherche en informatique pour développer un réseau d?ordinateurs interconnectés et naissance du Galactic Network, première communauté sociotechnique, par J.C.R Licklider ? premier projet open-source

· 1969 : le noyau UNIX est mis au point

· 1972 : première démonstration d?ARPANET

· 1976 : duplication d?un interpréteur Basic pour ALtair

· 1977 : BSD (Berkeley Software Distribution) voit le jour avec un ensemble de logiciels UNIX

· 1979 : le système en réseau de forum Usenet est inventé

- 1981 : IBM ouvre son architecture de CPU
- 1984 : Richard Matthew Stallman crée la GNU, un système d?exploitation libre
- 1985 : R.M.Stallman crée la Free Software Foundation, une organisation à but non-lucratif dont le rôle est de promouvoir et d?encourager l?utilisation du logiciel libre
- 1991 : Linus Torvalds annonce la naissance de Linux
- 1993 : la société Red Hat développe sa distribution basée sur Linux destinée aux entreprises et la distribution Debian est créée
- 1994 : premiers développements de MySQL, un système de gestion des bases de données pour sites Web dynamique
- 1995 : le système Apache, application pour serveurs Web voit le jour
- 1998 : création de la fondation Mozilla
- 2003 : naissance de Wikimedia qui mettra au point le projet Wikipedia
- 2004 : déploiement d?Ubuntu par Canonical, distribution destinée au grand public

6 | Rétro-ingénierie et sécurité

L?intégrité d?un logiciel concerne d?abord le logiciel en lui-même. Cela consiste, par exemple, à protéger le secret industriel contenu dans un code source, mais également les fonctionnalités offertes par le logiciel.

Mais, la captivité du logiciel dans sa forme compilée ou dans le matériel en lui-même n?assure aucune protection.

Des outils existent pour retrouver un code source intelligible par l?humain à partir de la version compilée. Certains langages modernes, avec .NET, par exemple, permettent même une décompilation redonnant le code source tel qu?il était à l?origine, ou en tout cas dans le même langage qu?à l?origine. La décompilation se résume finalement à une traduction entre deux langages : l?un intelligible par la machine et l?autre par l?humain.

La captivité dans le matériel demande une étape supplémentaire qui consiste à extraire le logiciel pour le décompiler. Lorsque ce n?est pas possible, l?étude poussée du matériel permet d?en détecter les failles.

La seule différence réside dans le temps nécessaire pour accéder au code source, pas dans la possibilité ou pas d?y avoir accès. L?empilement de couches d?enfermement (compilation, matériel, vernis noir sur le matériel) induit, en prime, de multiples sources de problèmes de sécurité du fait des multiples couches.

Il est possible de trouver ici [8] un exemple de décompilation.

7 | Sécurité et popularité

Il semble curieusement admis que Windows n?est pas sécurisé et que Linux l?est beaucoup plus. Dans la pratique, même si les techniques de développement donnent effectivement un niveau de sécurité différent, l?origine de la multiplication des failles logicielles provient pour bonne part de la popularité de la plateforme utilisée. En pratique, un logiciel utilisé par deux personnes dans le monde semblera plus sécurisé que l?autre utilisé par 2 millions de personnes pour la simple raison que moins de personnes iront chercher des failles dans le logiciel utilisé par 2 personnes. A titre d?illustration, Windows serait utilisé par 82,5% des utilisateurs alors que Linux n?en compterait que 5% (**http://www.w3schools.com/browsers/browsers_os.asp** [9]).

Les cibles potentielles sur Windows sont donc 16 fois plus élevées que sur Linux. Par ailleurs, Linux requiert certaines connaissances et donc une sensibilité plus importante à l?informatique. Ainsi, attaquer des machines sous Linux, à grande échelle, n?a que très peu d?intérêt.

Dans la pratique, même sur un logiciel non disponible sur le marché, une personne motivée pourra aller y chercher des failles de sécurité et les exploiter. La sécurité effective est donc relative à la popularité. La sécurité réelle est, elle, totalement indépendante.

8 | Open-source et communauté de développement

La réussite d?un logiciel open source dépend exclusivement de sa communauté. Sans implication de la communauté, il n?y a rien. Si la communauté est déséquilibrée dans sa composition par exemple, s?il y a beaucoup plus d?utilisateurs que de développeurs, alors les différentes mises à jour et les patchs mettront plus de temps à être conçus, ce qui peut avoir pour effet de compromettre la sécurité.

D?autre part, pour constituer une communauté, il faut des volontaires, intéressés par le but du logiciel, ce qui n?est pas toujours en adéquation avec les buts recherchés par les entreprises. Les prises de décision des dirigeants de la communauté sont parfois remises en question, il peut y avoir des forks [1], entraînant une probable mort lente d?un des deux projets. Ceci dit, si un projet open source meurt, c?est lentement, ce qui laisse du temps pour trouver une alternative. Il peut aussi y avoir une absorption des développeurs majoritaires et création de fork.

Enfin, au-delà de la question des utilisateurs et des développeurs, se pose la question des autres compétences. Il est nécessaire d?avoir également des graphistes, des traducteurs, des rédacteurs pour que le projet se développe, se répande et gagne en visibilité.

Il y a donc un travail de visibilité à fournir sur l?open-source ainsi que de démocratisation des usages. A titre d?exemple, faire en sorte que les logiciels libres et open-source soient utilisés dans les établissements scolaires, en particulier dans les cours d?informatique, afin que les élèves ne soient pas limités à l?utilisation d?un seul type d?outils.

[1] Le fork est un nouveau logiciel ou un nouveau système d?exploitation créé à partir d?un code source existant

Crédit photo : Ian Lishman/MASTERFILE

 648-03515359n.jpg [10]
Document de travail
0

URL source: http://labs.hadopi.fr/wikis/synthese-du-workshop-open-source-et-securite

Liens:
[1] http://labs.hadopi.fr/users/tris
[2] http://labs.hadopi.fr/wikis?tags=logiciel libre
[3] http://labs.hadopi.fr/wikis?tags=open-source
[4] http://labs.hadopi.fr/wikis?tags=sécurité
[5] http://labs.hadopi.fr/actualites/workshop-open-source-et-securite
[6] https://www.gnu.org/philosophy/categories.fr.htm
[7] http://www-ipst.u-strasbg.fr/pat/internet/histinfo/rfprj.htm
[8] http://paste.dprogramming.com/dpbbfgco
[9] http://www.w3schools.com/browsers/browsers_os.asp
[10] http://labs.hadopi.fr/sites/default/files/wiki/5025/image/648-03515359n.jpg