autor opracowania: Krzysztof Jurewicz

wersja I 09/08/2016

NASK

LABORATORIUM EE

Wykorzystanie wolnych licencji w projektach rządowych

Niniejszy dokument rozpatruje wykorzystanie wolnych licencji oprogramowania w projektach rządowych, poprzez przedstawienie stanowisk wybranych organów wybranych państw w wyżej wymienionych kwestiach oraz dokonanie syntezy tych stanowisk.

SPIS TREŚCI

1. Synteza Opracowania i uwagi dodatkowe

1. Kod otwarty nie jest sam z siebie bardziej lub mniej bezpieczny niż zamknięty kod własnościowy.

2. Ukrycie kodu nie czyni tego kodu bezpieczniejszym per se. Może ono utrudnić dotarcie do informacji o lukach, ale nie spowoduje, że te luki znikną.

3. Otwarcie kodu pomaga nie tylko atakującym, ale również zabezpieczającym kod, pozwalając społeczności na wykrycie luk, które częstokroć zostałyby przeoczone przez wąski zespół głównych programistów.

4. Skuteczny atak wymaga, aby w danym momencie o luce w kodzie produkcyjnym wiedzieli tylko atakujący (w p.p. zostałaby ona załatana). Upublicznienie kodu oraz ustanowienie vacatio legis przed jego wdrożeniem może zmniejszyć taką możliwość.

5. Zabezpieczanie oprogramowania przez ukrywanie informacji o lukach w nim jest metodą podrzędną względem tworzenia bezpiecznego oprogramowania jako takiego. O ile bezpieczeństwo oprogramowania per se **może być obiektywnie** analizowane, to nigdy nie można mieć pewności co do tego, kto posiada informacje o istniejących lukach w systemie.

6. Ukrywanie kodu może stanowić pokusę dla programistów, aby nie zabezpieczać go tak dobrze, jak kod publiczny.

7. Otwarcie kodu może obniżać koszty jego utrzymania, dając możliwość wnoszenia weń wkładu różnym zewnętrznym podmiotom.

8. Zamknięcie kodu rodzi pytania o to, dlaczego system finansowany z podatków ma nie być otwarty dla podatników. Abstrahując od pozostałych kwestji, wydaje się, że co do zasady systemy informatyczne finansowane z podatków powinny być otwarte, natomiast zamknięcie kodu mogłoby następować przy wystąpieniu ważnych przyczyn (np. niejawność projektu).

9. Otwarcie kodu zwiększa przejrzystość działania państwa i ułatwia audyt wydatkowania pieniędzy publicznych.

10. Programy typu bug bounty pozwalają zachęcić społeczność do znajdowania błędów w publicznym kodzie (co w efekcie prowadzi do ich załatania). W przypadku kodu zamkniętego korzyści ze znajdowania luk osiągają głównie osoby mające niecne zamiary.

11. Otwarcie kodu ułatwia tworzenie i utrzymywanie otwartych standardów.

2. Stanowiska Poszczególnych Państw

2.1. Wielka Brytania

Dostępna jest notatka „otwarte oprogramowanie a bezpieczeństwo" z 2011 r., wydana przez Home Office oraz Cabinet Office. W pkt. 1 mówi ona: „Otwarte oprogramowanie, jako kategoria, nie jest ani bardziej ani mniej bezpieczne niż zamknięte oprogramowanie własnościowe.".

2.2. Australia

Rząd Australijski w 2011 r. opublikował wytyczne odnośnie oprogramowania otwartoźródłowego, m.in. nakazujące agencjom rządowym traktować oprogramowanie otwartoźródłowe na równi z oprogramowaniem zamkniętoźródłowym. Dostępny jest też przewodnik po oprogramowaniu otwartoźródłowym. Sekcja piąta w/w dokumentu porównuje oba rodzaje oprogramowania.

2.3. Stany Zjednoczone

POLITYKA FEDERALNA

Biały Dom ma naszkicowaną federalną politykę kodu źródłowego. Można w niej przeczytać co następuje:

> While the benefits of enhanced Federal code reuse are significant, additional benefits can accrue when code is also made available to the public as Open Source Software (OSS). Making code available with an OSS license can enable continual improvement of Federal code projects when a broader community of users implements the code for its own purposes and publishes bugs and improvements. A number of private sector companies have already shifted some of their software development projects to an open source model, in which the source code of the software is made broadly available to the public for inspection, improvement, and reuse. In fact, several Federal agencies and component organizations also have already begun publishing custom-developed code under open source licenses or in the public domain, as discussed further below. Moreover, the Administration made a commitment, as part of its Second Open Government National Action Plan, to develop an Open Source Software policy that, together with the U.S. Digital Services Playbook, will support improved access to custom code developed for

> the Federal Government. This policy fulfills that commitment in an effort to improve U.S. Government software development and make the Government more open, transparent, and accessible to the public. Just as the Administration's Open Data Policy contributed to the creation of valuable and successful private businesses and services based upon open data released by the Government, improving access to taxpayer-funded source code can help facilitate similar results predicated on OSS.

Jeden z celów tej polityki to ustanowienie „establishing requirements for releasing code in the public domain or as OSS, including requirements for covered agencies to secure the rights necessary to make some custom-developed source code releasable to the public as OSS". Polityka ta nie ma jednak zastosowania do Narodowych Systemów Bezpieczeństwa.

DEPARTAMENT OBRONY

Departament Obrony USA przygotował stronę zawierającą najczęściej zadawane pytania na temat oprogramowania otwartoźródłowego. Dokument jest dość wyczerpujący; wybrane pytania poniżej.

> HOW DOES OPEN SOURCE SOFTWARE WORK WITH OPEN SYSTEMS/OPEN STANDARDS?
>
> Open standards can aid open source software projects:
>
> [...]
>
> Note that open standards aid proprietary software in exactly the same way.
>
> OSS aids open standards, too:
>
> - OSS implementations can help create and keep open standards open. A FLOSS implementation can be read and modified by anyone; such implementations can quickly become a working reference model (a „sample implementation" or an „executable specification") that demonstrates what the specification means (clarifying the specification) and demonstrating how to actually implement it. Perhaps more importantly, by forcing there to be an implementation that others can examine in detail, resulting in better specifications that are more likely to be used.
> - OSS implementations can help rapidly increase adoption/use of the open standard. OSS programs can typically be simply downloaded and tried out, making it much easier for people to try it out

and encouraging widespread use. This also pressures proprietary implementations to limit their prices, and such lower prices for proprietary software also encourages use of the standard.

With practically no exceptions, successful open standards have OSS implementations.

So, while open systems/open standards are different from open source software, they are complementary and can work well together.

DOES THE DOD USE OSS FOR SECURITY FUNCTIONS?

Yes. The 2003 MITRE study, „Use of Free and Open Source Software (FOSS) in the U.S. Department of Defense", for analysis purposes, posed the hypothetical question of what would happen if OSS software were banned in the DoD, and found that OSS „plays a far more critical role in the DoD than has been generally recognized… (especially in) Infrastructure Support, Software Development, Security, and Research". In particular, it found that DoD security „depends on (OSS) applications and strategies", and that a hypothetic ban „would have immediate, broad, and in some cases strongly negative impacts on the ability of the DoD to analyze and protect its own networks against hostile intrusion. This is in part because such a ban would prevent DoD groups from using the same analysis and network intrusion applications that hostile groups could use to stage cyberattacks. It would also remove the uniquely (OSS) ability to change infrastructure source code rapidly in response to new modes of cyberattack".

DOESN'T HIDING SOURCE CODE AUTOMATICALLY MAKE SOFTWARE MORE SECURE?

No. Indeed, vulnerability databases such as CVE make it clear that merely hiding source code does not counter attacks:

- Dynamic attacks (e.g., generating input patterns to probe for vulnerabilities and then sending that data to the program to execute) don't need source or binary. Observing the output from inputs is often sufficient for attack.
- Static attacks (e.g., analyzing the code instead of its execution) can use pattern-matches against binaries – source code is not needed for them either.
- Even if source code is necessary (e.g., for source code analyzers),

adequate source code can often be regenerated by disassemblers and decompilers sufficiently to search for vulnerabilities. Such source code may not be adequate to cost-effectively maintain the software, but attackers need not maintain software.

- Even when the original source is necessary for in-depth analysis, making source code available to the public significantly aids defenders and not just attackers. Continuous and broad peer-review, enabled by publicly available source code, improves software reliability and security through the identification and elimination of defects that might otherwise go unrecognized by the core development team. Conversely, where source code is hidden from the public, attackers can attack the software anyway as described above. In addition, an attacker can often acquire the original source code from suppliers anyway (either because the supplier voluntarily provides it, or via attacks against the supplier); in such cases, if only the attacker has the source code, the attacker ends up with another advantage.

Hiding source code does inhibit the ability of third parties to respond to vulnerabilities (because changing software is more difficult without the source code), but this is obviously not a security advantage. In general, “Security by Obscurity” is widely denigrated.

This does not mean that the DoD will reject using proprietary COTS products. There are valid business reasons, unrelated to security, that may lead a commercial company selling proprietary software to choose to hide source code (e.g., to reduce the risk of copyright infringement or the revelation of trade secrets). What it does mean, however, is that the DoD will not reject consideration of a COTS product merely because it is OSS. Some OSS is very secure, while others are not; some proprietary software is very secure, while others are not. Each product must be examined on its own merits.

WHAT ARE INDICATORS THAT A SPECIFIC OSS PROGRAM WILL HAVE FEWER UNINTENTIONAL VULNERABILITIES?

As noted in the Secure Programming for Linux and Unix HOWTO, three conditions reduce the risks from unintentional vulnerabilities in OSS:

1. Developers/reviewers need security knowledge. Knowledge is more important than the licensing scheme.
2. People have to actually review the code.

3. This has a reduced likelihood if the program is niche/rarely-used, few developers, rare computer language, or not really OSS. Conversely, if it widely-used, has many developers, and so on, the likelihood of review increases. Examine if it is truly community-developed – or if there are only a very few developers.
4. Review really does happen. Several static tool vendors support analysis of OSS (such as Coverity and Fortify) as a way to improve their tools and gain market use. There are many general OSS review projects, such as those by OpenBSD and the Debian Security Audit team. And of course, individual OSS projects often have security review processes or methods (such as Mozilla's bounty system). If there are reviewers from many different backgrounds (e.g., different countries), this can also reduce certain risks. When examining a specific OSS project, look for evidence that review (both by humans and tools) does take place.
5. Problems must be fixed. It is far better to fix vulnerabilities before deployment – are such efforts occuring? When the software is already deployed, does the project develop and deploy fixes?

DOES THE DOD ALREADY USE OPEN SOURCE SOFTWARE?

Yes, extensively. The 2003 MITRE study, „Use of Free and Open Source Software (FOSS) in the U.S. Department of Defense", identified some of many OSS programs that the DoD is already using, and concluded that OSS „plays a more critical role in the [Department of Defense (DoD)] than has generally been recognized".

[...]

IS THE GPL COMPATIBLE WITH GOVERNMENT UNLIMITED RIGHTS CONTRACTS, OR DOES THE REQUIREMENT TO DISPLAY THE LICENSE, ETC, VIOLATE GOVERNMENT UNLIMITED RIGHTS CONTRACTS?

The GPL and government „unlimited rights" terms have similar goals, but differ in details. This isn't usually an issue because of how typical DoD contract clauses work under the DFARS.

Any software that has a non-government use and is licensed to the public is commercial software, by definition, including OSS programs licensed to the government using the GPL. Normally the government only expects to get the usual commercial rights to commercial software, and not „unlimited rights". So if the software displays a license in a way that can't be legally disabled (as required by the GPL), there is no problem, because this is an ordinary commercial software license term. The same would be true if you used Microsoft Windows; you aren't normally permitted to disable the rights-display functions of Microsoft Windows either.

In contrast, the government normally gets „unlimited rights" only when it pays for development of that software, in full or in part. Software developed by government funding would typically be termed „noncommercial software", and thus falls under different rules. The government does have the right to take software it has unlimited rights to, and link it with GPL software. After all, the government can use unlimited rights software in any way it wishes.

Once the government has unlimited rights, it can release that software to the public in any it wishes – including by using the GPL. This is not a contradiction; it's quite common for different organizations to have different rights to the same software. The program available to the public may improve over time, through contributions not paid for by the U.S. government. In that case, the U.S. government can choose to use the version to which it has unlimited rights, or it can use the publicly-available commercial version available to the government through that version's commercial license (the GPL in this case).

HAS THE U.S. GOVERNMENT RELEASED OSS PROJECTS OR IMPROVEMENTS?

Yes, both entirely new programs and improvements of existing OSS. There are far too many examples to list; a few examples are:

- Delta3D
- Security-Enhanced Linux (SELinux)
- OpenVista
- Expect
- EZRO
- Evergreen (by the State of Georgia),
- OpenSSL (this improvement was a Common Criteria evaluation)
- Bind implementation of DNSSEC
- GNAT Ada compiler
- BSD TCP/IP suite

WHAT ARE THE RISKS OF THE GOVERNMENT NOT RELEASING SOFTWARE AS OSS?

If the government modifies existing OSS, but fails to release those improvements back to the main OSS project, it risks:

- Greatly increased costs, due to the effort of self-maintaining its own version
- Inability to use improvements (including security patches and innovations) by others, where it uses a „non-standard" version instead of the version being actively maintained

Similarly, the government develops runs the following risks when it develops new software but does not release it as OSS, it risks:

- Greatly increased cost, due to having to bear the entire burden of development costs
- Inability to use improvements (including security patches and innovations) by others, since they do not have the opportunity to aid in its development
- The development and release of a competing OSS project. In this case, the government has the unenviable choice of (1) spending possibly large sums to switch to the OSS project (which would typically have a radically different interface and goals), or (2) continuing to use the government-unique custom solution, leaving the U.S. systems far less capable that others' (including our adversaries)
- Questions about why the government – who represents „the people" – is not releasing software that they paid for back to „the people".

Clearly, classified software cannot be released back to the public as open source software. However, often software can be split into various components, some of which are classified and some of which are not, and it is to these unclassified portions that this text addresses.

WHAT ARE THE RISKS OF THE GOVERNMENT RELEASING SOFTWARE AS OSS?

The key risk is the revelation of information that should not be released to the public. Classified software should already be marked as such, of course. This risk is mitigated by reviewing software (in particualr, for classification and export control issues) before public release.

CAN GOVERNMENT EMPLOYEES DEVELOP SOFTWARE AND RELEASE IT UNDER AN OPEN SOURCE LICENSE?

Not under typical open source software licenses based on copyright, but there is an alternative with the same practical effect.

Software developed by US federal government employees (including military personnel) as part of their official duties is not subject to copyright protection and is considered “public domain” (see 17 USC § 105). Public domain software can be used by anyone for any purpose, and cannot be released under a copyright license (including typical open source software licenses).

However, software written entirely by federal government employees as part of their official duties can be released as “public domain” software. This is not under a copyright license, it is absence of a license. By some definitions this is technically not an open source license (because no license is needed), but “public domain” software can be legally used, modified, and combined with other software without restriction. Thus, “public domain” software provides recipients all of the rights that open source software must provide. An example of such software is Expect, which was developed and released by NIST.

Government employees may also modify existing open source software. If some portion of the software was developed by persons who are not US government employees, then the software can be released under copyright license. (See next question.)

[...]