

Łukasz Karcz

## OPEN SOURCE ABSOLUTNIE OD PODSTAW

Zarówno w Europie Zachodniej, jak i w Polsce, coraz więcej mówi się o Wolnym Oprogramowaniu [OpenSource], konieczności wdrażania Linuksa i kwestiach licencyjnych. Nie sposób jednak nie spostrzec, że do dyskusji tej wkrada się często sporo przeinaczeń, błędów czy wręcz krańcowego niezrozumienia meritum sprawy, czego przykładem jest chociażby niedawna wypowiedź Pani Hanny Kuzińskiej z 9 stycznia 2004. Głównym powodem całego zamieszania jest fakt, że kwestia decyzji o wdrożeniu oprogramowania OpenSource dość niespodziewanie spada na osoby, które z racji swoich uprawnień i prerogatyw nie posiadały w tym zakresie wystarczającej wiedzy, a całość spraw związanych z zakupem sprzętu i oprogramowania informatycznego cedowały dotychczas na odpowiednio wykwalifikowanych pracowników. Podjęcie decyzji o migracji w kierunku Wolnego Oprogramowania wiąże się dla nich często z niezrozumieniem, czym jest owo Wolne Oprogramowanie i jakie konsekwencje niesie dla firm czy instytucji migracja w stronę tej formy własności intelektualnej. Napisałem ten artykuł właśnie z myślą o osobach, które w swojej pracy mają niewielką styczność z powyższymi kwestiami, nie mając wykształcenia i doświadczenia informatycznego, a stoją bądź niedługo staną wobec podjęcia decyzji mającej wpływ na kształt systemu informatycznego.

Pisząc artykuł do osób nie będących informatykami, nie sposób było ustrzec się pewnych uproszczeń, uciekając w wyjaśnieniach od zaciemniających sformułowań technicznych czy też spływając niektóre kwestie. Z tego też powodu niniejsze opracowanie zawiera dość dużą porcję analogii czy uproszczeń, które momentami mogą wydać się zbyt przejawskawione czy wręcz trywialne dla osób zawodowo zajmujących się programowaniem. Stojąc jednak na rozdrożu pomiędzy możliwie jak najprostszym przedstawieniem tematu a zagłębieniem się w szczegóły techniczne, wybrałem pierwszą drogę mając nadzieję, że osoby zainteresowane tematyką bez trudu znajdą w Sieci materiały dla pogłębienia swojej wiedzy.

W niniejszym artykule omawiam istotę programu komputerowego, objaśniam, czym są źródła programu, a także definiuję pojęcie Wolnego Oprogramowania. W dalszej części artykułu przedstawiam zalety i wady korzystania z oprogramowania dostępnego na zasadzie OpenSource. Zakładam, że Czytelnik posiada podstawowe informacje z zakresu pracy w środowisku Windows, dlatego też część porównań i przykładów odnosi się właśnie do tego środowiska. Gdzieś tam podałbym konkretne przykłady, na których można sprawdzić podane informacje – ostrzegam jednak, że nieumiejętne sprawdzenie przykładów może doprowadzić do niemożności uruchomienia niektórych programów, a w skrajnych przypadkach do zawieszenia systemu czy utraty ważnych danych.

W artykule używam zamiennie pojęć „Wolne Oprogramowanie” „Open Source” i „Free Software”.

### Program komputerowy i jego źródła

Zanim przejdę do meritum i zdefiniowania pojęcia Wolnego Oprogramowania, wyjaśnię pokrótce, czym tak naprawdę [w kontekście naszego artykułu] jest program komputerowy.

Program komputerowy, zwany czasem również aplikacją, to z punktu widzenia użytkownika jeden lub więcej plików komputerowych, podobnie jak np. dokument w edytorze tekstowym. Jest to jednak plik o tyle specyficzny, że po jego otwarciu nie uzyskujemy dostępu do żadnych danych, a właśnie uruchamiamy ten program. Na dysku twardym komputera zarówno dokument zawierający ofertę handlową, jak i arkusz kalkulacyjny z cennikiem czy program zapisywane są w postaci jednego lub więcej plików. Nie ma między nimi niemal żadnej różnicy [oczywiście poza zawartością i faktem, że programy można uruchamiać, a inne pliki nie] – programy i inne pliki mogą tak samo wysyłać pocztą elektroniczną, usuwać, kopiować i zmieniać nazwy. Ma to oczywiście wpływ na ich działanie i stanowczo odradzam zmianę ich nazw bądź ich usuwanie – jednak z punktu widzenia tego artykułu pliki programów i pliki dokumentów są pod pewnymi względami bardzo do siebie podobne.

Program komputerowy sam w sobie zawiera instrukcje dla komputera, które ten wykonuje w momencie uruchomienia programu. Bardzo ważną rzeczą dla zrozumienia dalszej części artykułu jest świadomość, że **w programie komputerowym wykonywanym przez użytkownika** [czyli np. w momencie uruchomienia dowolnej gry czy arkusza kalkulacyjnego] **instrukcje dla komputera**

**zapisane są w postaci kompletnie nieczytelnej dla człowieka.** Gdyby otworzyć dowolny program [pliki programów mają rozszerzenie .EXE bądź .COM] np. w Notatniku czy WordPadzie, zobaczylibyśmy kompletnie nieczytelny zapis krzaczków i przypadkowy ciąg liter i cyfr. Jest to jednak postać czytelna dla komputera i właśnie poprzez odczyt tych „krzaczków” komputer „wie”, że ma na ekranie narysować ramkę, wyświetlić przycisk czy posumować komórki w naszej liście płac, którą przed momentem stworzyliśmy. Dla użytkownika nie ma to jednak znaczenia – otwierając np. arkusz kalkulacyjny nie interesuje nas zawartość pliku EXE. Wydajemy po prostu komputerowi polecenie uruchomienia danego programu czy gry, a komputer sam robi resztę.

Błędem jednakże byłoby sądzić, że te „krzaczki” są **bezpośrednim** efektem pracy programistów. Plik .EXE [zwany również plikiem wykonywalnym] powstaje jako efekt przełożenia kodu zrozumiałego dla człowieka na język maszyny przy pomocy tzw. kompilatora. Typowy ciąg czynności, jaki wykonywany jest przez programistę to cykl: napisanie kodu – kompilacja [czyli przełożenie kodu na język komputera], w efekcie czego powstaje właśnie ów plik programu, czyli np. edytor tekstów, najnowsza gra, czy arkusz kalkulacyjny. Po kompilacji następuje jeszcze linkowanie, czyli dołączenie wcześniej stworzonych modułów, ale na razie dla naszych rozważań najważniejszy jest właśnie duet pisanie – kompilacja.

W tym momencie dochodzimy do kluczowego dla artykułu pojęcia **kodu źródłowego**. Tym terminem określa się właśnie kod pisany przez programistę, na podstawie którego tworzony jest plik programu.

Nie wdając się w szczegóły, fragment struktury kodu tworzonego przez programistę może wyglądać np. tak, jak poniżej. Przykład jest oczywiście abstrakcyjny, celowo nie napisałem go w żadnym konkretnym języku, żeby nie zaciemniać artykułu wyjaśnieniami dotyczącymi składni, czy stosowanych języków programowania:

```
otwórz_okno_programu;  
dodaj_menu_do_programu;  
narysuj_obrazek;  
...
```

Plik lub pliki zawierające tego typu instrukcje „przepuszcza się” przez kompilator / linker, otrzymując w efekcie gotowy do użytku program komputerowy. Oczywiście w „prawdziwym” programie instrukcje te są inne i jest ich znacznie więcej. Chodzi o to, żeby pokazać relację między kodem źródłowym, a plikiem .EXE.

Dobłą analogią do zastosowania w tym miejscu jest porównanie do ciasta, przepisu i składników. Podobnie, jak na podstawie przepisu i składników każdy, kto dysponuje odpowiednią wiedzą i umiejętnościami, może zrobić dowolne ciasto, tak samo dysponując kodami źródłowymi można odtworzyć program, a także dokonać w nim dowolnej zmiany. Wystarczy po prostu dopisać / usunąć zmiany i przekompilować program. W ten sposób możnaby dodać nową funkcję do gier, dopisać nową funkcję programu czy usunąć błąd, który powoduje np. zawieszenie programu. Albo sprawić, żeby jakieś okno otwierało się w innym miejscu.

Możnaby... gdybyśmy mieli dostęp do kodów źródłowych, prawda?

## **Oprogramowanie otwarte i zamknięte**

I tutaj dochodzimy do sedna sprawy. Spora część oprogramowania w dalszym ciągu dostarczana jest **bez kodów źródłowych**. Kupując w sklepie komputerowym grę, edytor tekstów czy program do fakturowania dostajemy program. Możemy go uruchamiać, korzystać z niego, zarabiać przy jego pomocy na życie... i na tym nasze prawa w większości się kończą. Jeśli wczytamy się w umowę licencyjną dla takiego programu, bez trudu dostrzeżemy w niej zakaz jego rozpowszechniania czy dokonywania jakichkolwiek modyfikacji w programie [modyfikacja programu bez kodów źródłowych, acz dość trudna, nie jest niemożliwa].

Na drugim końcu skali znajduje się zdobywające coraz większą popularność oprogramowanie typu „Open Source”, czyli oprogramowanie o otwartych źródłach. Oprogramowanie takie dostępne jest razem ze źródłami, a więc razem z gotowym programem dostajemy także kody źródłowe, dzięki którym

możemy swobodnie modyfikować kod dla własnych potrzeb, dopisywać nowe możliwości i poprawiać dostrzeżone błędy.

Oprogramowanie o otwartych źródłach rozpowszechniane jest zazwyczaj na licencji GNU GPL [General Public Licence]. Podstawowym wymogiem tej licencji jest zapis, że każdy program wykorzystujący otwarty kod i publicznie udostępniony musi również udostępnić swoje źródła. Jeśli więc korzystam z jakiejś funkcji zawartej w kodzie udostępnionym na zasadach OpenSource, pisząc swój program, to jeśli chcę go upublicznić [bezpłatnie bądź odpłatnie], muszę również udostępnić do niego źródła.

„Open Source” vel „Free Software” zawiera małą pułapkę językową. Angielski przymiotnik *free* oznacza tu bowiem nie *darmowy*, ale właśnie *wolny*. Określenie jest nieco mylące, ponieważ oprogramowanie tego typu dostarczane jest w sporej większości za darmo, ale nie to stanowi jego nadrzędną cechę. Podstawową cechą tworu o nazwie „Free Software” jest właśnie *wolność* do jego rozpowszechniania, *wolność* dostępu do kodów źródłowych i *wolność* jego modyfikacji.

Programy oparte o zasadę wolności źródeł mogą być także tworzone na indywidualne zamówienie. Nic nie stoi na przeszkodzie [poza dobrą wolą obu stron], żeby firma bądź instytucja zleciła podwykonawcy stworzenie oprogramowania, którego kody źródłowe zostaną następnie przekazane zleceniodawcy. Wolność źródeł – co wielokrotnie podkreślają autorzy licencji GNU GPL – nie wyklucza w żaden sposób czerpania z nich korzyści majątkowych. Zleceniodawca – jeśli tylko nie przekazuje w żaden sposób gotowego programu – nie ma obowiązku również przekazywania dalej źródeł kodu.

### Oprogramowanie OpenSource jako produkt handlowy

Typowa „paczka” oprogramowania wypuszczonego na zasadzie OpenSource składa się z kodów programu, elektronicznej dokumentacji, kopii wspomnianej licencji GNU GPL i programów już skompilowanych, gotowych do uruchomienia i użytku. Tutaj też otwiera się pole do zarobku dla firm chcących oprogramowanie typu OpenSource włączyć do własnej oferty handlowej. Trudno przecież liczyć na to, że ktoś będzie chciał kupić oprogramowanie, które w sporej części może za darmo ściągnąć z Internetu. A jednak spora część firm istnieje i ma się zupełnie dobrze, żyjąc ze sprzedaży takiego właśnie oprogramowania.

Jak to możliwe? Otóż mimo faktu, że program jest zazwyczaj powszechnie dostępny, nie wszyscy mają czas i ochotę zajmować się jego utrzymaniem, konserwacją, szkoleniem użytkowników itp. Na oprogramowaniu OpenSource można zarabiać na kilka sposobów. Wśród nich wymienić można m.in.:

- oferowanie odpłatnej pomocy technicznej,
- sprzedaż programu w rozszerzonej wersji – z dołączoną drukowaną instrukcją czy dodatkowym oprogramowaniem,
- szkolenia oferowane dla użytkowników,
- instalacje i utrzymywanie działających systemów u klienta.

### OpenSource – zmiana sposobu myślenia o tworzeniu oprogramowania

Najbardziej widoczną częścią świata OpenSource, mimo wspomnianej przeze mnie możliwości tworzenia oprogramowania „pod klucz”, są aplikacje z kodem źródłowym upublicznionym w Internecie. Programy te rozwijają się zazwyczaj bardzo szybko, a ich niezawodność zjednuje im liczne rzesze sympatyków, zniechęconych do często wadliwych i wbrew pozorom często pozbawionych opieki technicznej rozwiązań czysto komercyjnych, opartych na zamkniętych źródłach.

„Okrętami flagowymi” armady spod znaku Wolnego Oprogramowania są między innymi:

- **Linux** – bezpłatny system operacyjny z całkowicie dostępnymi źródłami; udoskonalany przez wiele lat, ewoluował z systemu przeznaczonego dla serwerów, gdzie liczy się niezawodność i zdolność do długiej, bezobsługowej pracy, do rangi systemu uniwersalnego, możliwego do wykorzystania również w biurach czy studiach graficznych,
- **OpenOffice** – darmowy pakiet biurowy, w niczym nie ustępujący komercyjnym rozwiązaniom oferowanym przez potentatów z branży,
- **Apache** – serwer WWW, czyli program odpowiedzialny za wysyłanie stron WWW do przeglądarki internetowej osoby odwiedzającej stronę,

- **Mozilla** – fantastyczna bezpłatna przeglądarka internetowa, m.in. praktycznie uniemożliwiająca zakażenie wirusami pochodzącymi z Internetu,
- **KDE/GNOME** – darmowe środowiska do pracy graficznej w systemie, będące swoistymi “nakładkami”, upodabniającymi Linuksa do znanego wyglądu Windows i oferującymi możliwości komfortowej pracy nawet osobom dopiero rozpoczynającym przygodę z komputerami.

Ktoś może spytać, dlaczego aplikacje te są rozwijane mimo wydawałoby się oczywistego faktu, że nikt nie korzysta bezpośrednio na ich rozwijaniu? Odpowiedzi jest kilka, wymienię te najpowszechniejsze:

- ekonomia: zamiast pisać od nowa program dla swoich potrzeb, znacznie taniej jest zaadoptować już istniejący,
- nauka: analiza i pisanie kodu ogólnodostępnego to świetna szkoła dla młodych adeptów informatyki,
- sława: kto nie chciałby zobaczyć swojego nazwiska na liście płac hollywoodzkiego hitu kasowego? Na podobnej zasadzie działa magia Wolnego Oprogramowania; dodatkowo udział w znanym projekcie to znakomita autoreklama chociażby podczas rozmowy o pracę...

## Bezpieczeństwo otwartego kodu

Dość częstym przytykiem, podnoszonym pod adresem rozwiązań OpenSource, jest kwestia jego bezpieczeństwa. Argumentuje się, że skoro oprogramowanie jest ogólnodostępne, to każdy może zajrzeć do środka i wyłapać błąd, który umożliwi mu np. włamanie na serwer internetowy obsługiwany przez wadliwie działający program.

Ile prawdy jest w powyższych stwierdzeniach? Otóż otwartość źródeł istotnie stwarza takie możliwości, jak opisano powyżej. Jest to jednak zaleta, a nie wada! Dlaczego? Chcąc odpowiedzieć na to pytanie, należy porównać oprogramowanie zamknięte z oprogramowaniem, którego kod dostępny jest publicznie:

1. Kupując oprogramowanie zamknięte, NIGDY nie mamy stuprocentowej pewności, czy producent nie umieścił w nim kodu, który np. wszystkie zaokrąglenia kwot faktur przeleje na jego konto [zdarzały się takie historie...], albo ostatniego dnia każdego miesiąca nie prześle naszych danych finansowych do jego siedziby przez Internet. Oprogramowanie otwarte, jakkolwiek nie wyklucza takiej możliwości, to jednak znacznie ją minimalizuje. W przypadkach skrajnych można po prostu przejrzeć podejrzany fragment kodu i sprawdzić, czy nie zawiera on poleceń, które w jakiś sposób są niebezpieczne dla naszych danych. Przypominając analogię z ciastkiem – kto nam da PEWNOŚĆ, że dane ciastko nie zostało doprawione arsenikiem, skoro nie mamy ani przepisu, ani nie możemy go upiec sami, a jedynie kupujemy go w cukierni?
2. Błędy w oprogramowaniu zamkniętym usuwane są czasem na żądanie klientów, czasem w momencie wypuszczenia nowej wersji programu, a czasem nigdy. W oprogramowaniu otwartym, błąd usuwany jest zazwyczaj bezpośrednio po jego wykryciu. A jeśli sami dysponujemy odpowiednio wykwalifikowanym personelem, możemy po prostu usunąć ten błąd sami.
3. Otwartość oprogramowania sprzyja pisaniu dobrych programów. Nikt trzeźwo myślący nie będzie narażał swojej opinii w środowisku programistycznym, wypuszczając źródła, z których wynika brak wiedzy, czy zwykłe niechlujstwo. Oprogramowanie zamknięte może być pisane dobrze, lecz nie musi – w końcu żaden klient nie zobaczy, w jaki sposób rozwiązaliśmy daną rzecz.

## Otwarty kod a otwarte dane

Mitem, który niestety często powiela się w mediach, jest obawa, że otwarcie kodu spowoduje wyciek na zewnątrz firmy poufnych danych. “Skoro przecież wszystko widać...” Należy sobie w tym miejscu uświadomić ważne rozróżnienie: jawność kodu ani nawet jawność sposobu, w jakim zostały zapisane dane [czyli tzw. algorytmów szyfrujących] nie ma **NIC WSPÓLNEGO** z jawnością samych danych. Z matematycznego punktu widzenia, nawet znając sposób, w jaki dane zostały zaszyfrowane, nie mamy możliwości odczytania poufnych danych, jeśli nie dysponujemy kluczami niezbędnymi do ich odczytania. To zaś, czy danym kluczem dysponujemy, czy nie, nie ma nic wspólnego z otwartością czy zamknięciem oprogramowania, a jedynie z odpowiednio prowadzoną polityką bezpieczeństwa firmy – dostęp do ważnych danych powinien być chroniony na podobnych zasadach, co dostęp do firmowej kasy, a nie poprzez ukrywanie źródeł programu.

Jeśli chcielibyśmy na uparteo stwierdzić istnienie JAKIEJKOLWIEK zależności między otwartością kodu a bezpieczeństwem danych, to jest to raczej zależność odwrotna. Posłużę się tutaj bardzo

proszym przykładem: kto nam zagwarantuje, że w jakimś programie z zamkniętymi źródłami programista nie umieścił np. hasła CHOINKA, po wpisaniu którego można będzie odczytać wszystkie nasze dokumenty? W oprogramowaniu otwartym wszystko, co jest robione w celu zapewnienia bezpieczeństwa naszym danym, można sprawdzić naocznie – a więc sprawdzić również, czy programista nie zostawił nieszczęsnego hasła CHOINKA zapisanego “na sztywno” w programie. W oprogramowaniu zamkniętym – niejako z definicji – możliwości sprawdzenia takiego zagrożenia nie ma i nie będzie.

## **Właściwy wybór, czyli korzyści z OpenSource dla zleceniodawcy**

OpenSource, przy wszystkich swoich zaletach, wydaje się być idealnym rozwiązaniem szczególnie dla dużych projektów informatycznych, gdzie podstawową rolę odgrywa bezpieczeństwo wdrożenia, a wdrażany system jest podstawowy dla działania firmy bądź organizacji. Na zakończenie artykułu pozwolę sobie przedstawić kilka zalet, wypływających z udostępnienia kodu zleceniodawcy:

1. Bezpieczeństwo danych.
2. Lepsza jakość kodu.
3. Możliwość modyfikacji przez użytkownika.
4. **Zwiększone bezpieczeństwo wdrożenia** – udostępnienie kodu gwarantuje zleceniodawcy niezależenie się od dostawcy oprogramowania w przypadku złej woli lub czynników trzecich [likwidacja firmy, zerwanie kontraktu, itp.] - mając źródła programów, modyfikujemy je we własnym zakresie lub po prostu przekazujemy je innej firmie jako podwykonawcy. Należy zauważyć, że wspomniany scenariusz nie umożliwi jakiegokolwiek wyłudzenia cudzej pracy – przekazanie kodu osobie trzeciej [czyli innemu podwykonawcy], jakkolwiek możliwe, wiąże się z koniecznością zapoznania się z kodem, znalezienia wykwalifikowanego personelu, etc. Możliwość przekazania kodu nie jest więc furtką dla wyłudzaczy, ile raczej wyjściem awaryjnym dla zleceniodawcy w sytuacji, gdy zleceniobiorca nie może lub nie chce kontynuować pracy nad projektem.

## **Coda**

Mam nadzieję, że niniejszy artykuł pomógł zrozumieć podstawy idei leżącej u podstaw myśli OpenSource. Rozwiązania oparte na Wolnym Oprogramowaniu doceniło już miliony firm [wśród nich tacy giganci, jak IBM, Novell czy Sun], a od jakiegoś czasu widać zainteresowanie nim ze strony urzędów państwowych w Europie Zachodniej. Otwarte Oprogramowanie jest forsowane przez Unię Europejską jako lepsza, bezpieczniejsza i przede wszystkim znacznie tańsza alternatywa dla rozwiązań komercyjnych. Pozostaje mieć więc nadzieję, że z biegiem czasu do rozwiązań wolnodostępnych przekonamy się także w Polsce.

W razie jakichkolwiek pytań można skontaktować się z autorem niniejszego artykułu, pisząc na adres torero (at) [poczta.onet.pl](mailto:poczta.onet.pl). W miarę możliwości postaram się odpowiedzieć na wszystkie pytania.

Do zobaczenia w świecie Wolnego Oprogramowania!

Łukasz Karcz