Watchdog Polska Opinia Ekspercka

opracowanie: mgr inż. Jarosław Żeliński

dla: Sieć Obywatelska Watchdog Polska

Opinia ekspercka w kwestii czy udostępnienia kodu źródłowego programu „System Losowego Przydziału Spraw" spowoduje wysokie ryzyko narażenia na skuteczne cyberataki na aplikację SLPS, na utratę poufności i integralności przetwarzanych danych jak i potencjalnie całej infrastruktury potrzebnej do jej działania w celu przedstawienia jej w toczącym się postępowaniu przed Wojewódzkim Sądem Administracyjnym (II SA/Wa 1785/22) ze skargi Zleceniodawcy na decyzję Ministra Sprawiedliwości w przedmiocie odmowy udostępnienia informacji publicznej.

1. Streszczenie

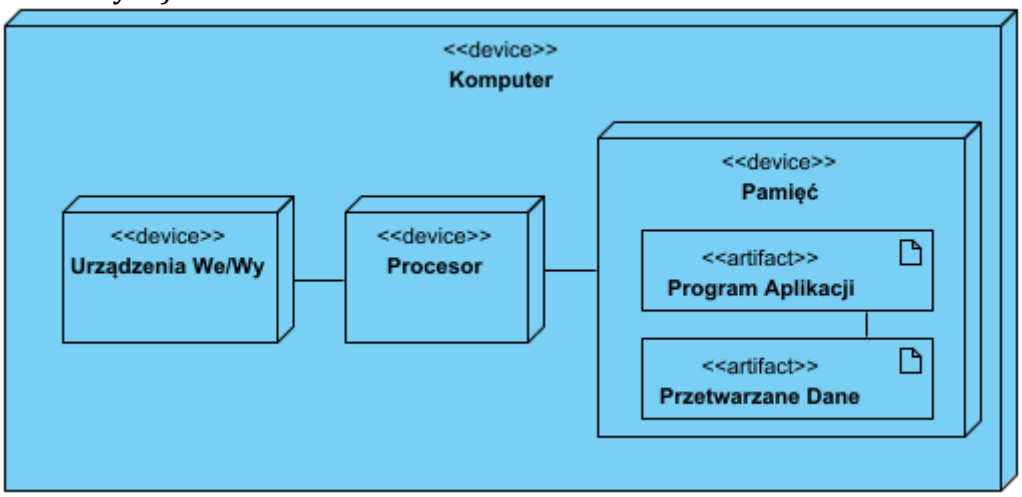
Kod źródłowy aplikacji, realizujący wymagania funkcjonalne wyrażone jako opis algorytmu, stanowi odrębną część oprogramowania. Część ta nie realizuje żadnych funkcji pozafunkcjonalnych, w szczególności takich jak bezpieczeństwo całego systemu. Kod stanowiący implementację tej części systemu, to kod wyrażający w innej formie dokładnie to zostało już opublikowane. W konsekwencji udostępnienie kodu źródłowego oprogramowania, rozumianego jak kodu komponentu realizującego funkcjonalność istotną dla Państwa i obywateli, stanowi istotny element statutowej działalności organizacji Watchdog Polska. Mimo tego, że nie ma takiego obowiązku, chcąc jednak zachować transparentność swoich działań i ich pobudek, uzyskany kod źródłowy i jego audyt, stanowią potwierdzenie, że realizuje on algorytm opisany w publicznych dokumentach, do sprawdzenia czego mamy prawo.

2. Metody i narzędzia

W toku opracowania ekspertyzy i wypracowania opinii na temat otrzymanego materiału użyto metod typowych dla publikacji naukowych to znaczy: wybrano narzędzia i metody, opracowano idealizację przedmiotu ekspertyzy (McMullin, 1985) i wypracowano wnioski na bazie porównania stanu faktycznego z idelizacją.
W tym przypadku autor oparł się na aktualnej wiedzy naukowej i doświadczeniu w podobnych projektach.

3. Aktualny stan wiedzy w badanym obszarze

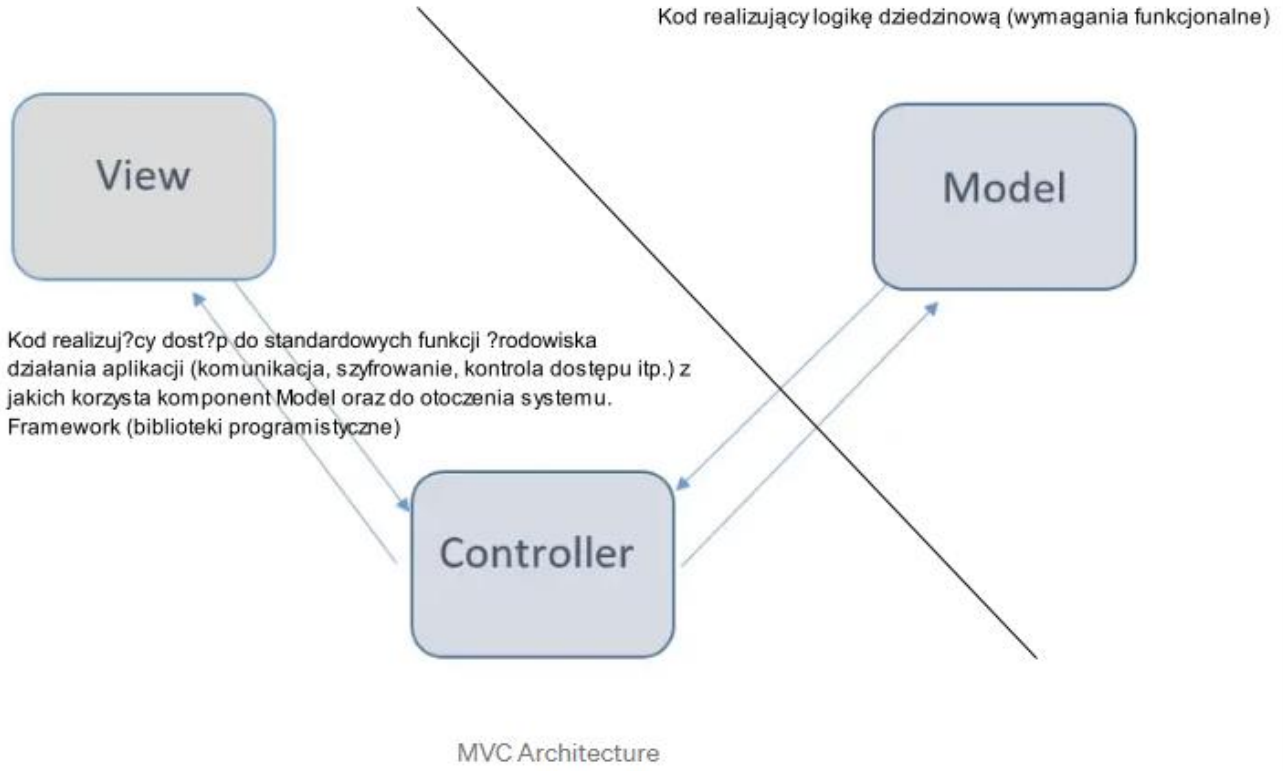
Kluczowym elementem pracy nad produktem jakiejkolwiek inżynierii jest projektowanie (słownik języka polskiego PWN: inżynieria to projektowanie i konstruowanie obiektów oraz urządzeń technicznych).



Architektura komputera

Komputer jest w filozofii informatyki określany jako jako "uniwersalny mechanizm". Określenie to jest istotne gdyż, oprogramowanie (kod) rozpatrujemy jako część komputera. Cechy użytkowe ma komputer a nie oprogramowanie, które jest jedynie zestawem poleceń dla procesora (Biernat, 1999).
Określenie mówiące, że komputer realizuje określony mechanizm dotyczy głównie programu komputerowego gdyż funkcjonalność komputera to wynik realizacji tego programu, ten niesie z sobą określone algorytmy (Murawski, 2015). Przedstawiono to schematycznie na diagramie Architektura komputera.

Warte odnotowania jest, że stwierdzenie "użytkownik używa programu" jest potocznym uproszczeniem, użytkownik używa komputera, w którym zainstalowano i uruchomiono określony program.



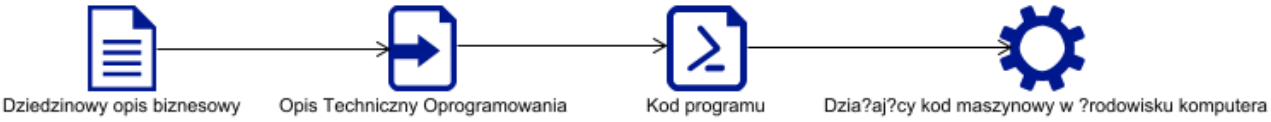
Wzorzec Architektoniczny MVC

Komponent Program Aplikacji zobrazowany na diagramie Architektura komputera to kod programu, ten ma zawsze określona architekturę (w najprostszym przypadku jest jednym monolitycznym blokiem).
MVC to skrót od Model-View-Controller. Istotą tego wzorca jest prezentacja architektury aplikacji (struktury kodu) w postaci trzech komponentów: Model to część kodu (komponent) realizująca dziedzinową logikę operacji na danych, Controller to część kodu, która odpowiada za połączenie Modelu z jego otoczeniem, zaś View to część kodu realizująca prezentację danych i formularzy na ekranie komputera (urządzenia wejścia wyjścia nazywanego graficznym interfejsem użytkownika). Tu należy podkreślić, że to co nazywamy potocznie kodem źródłowym oprogramowania, z reguły dotyczy wyłącznie części nazwanej Model.
Taka separacja jest często realizowana fizycznie (odrębne komponenty oprogramowania) lub tylko do celów analizy, gdzie stanowi jedynie abstrakcję opisanego podziału (aplikacje o monolitycznej architekturze). Jest to tu istotne, gdyż komponent Model realizuje wymagania funkcjonalne (to do czego komputer służy), zaś pozostałe komponenty realizują wymagania pozafunkcjonalne (jakość, bezpieczeństwo, ergonomia użytkowania itp.) (Svirca, 2020). Zobrazowano to na diagramie Wzorzec Architektoniczny MVC.
Należy tu także zwrócić uwagę na fakt, że kod realizujący pozafunkcjonalne wymagania (komponenty View i Controller) stanowi łącznie tak zwane środowisko wykonawcze aplikacji. Jest to tak zwany framework (lub runtime), stanowiący sobą zbiór gotowych bibliotek oprogramowania na wzór systemu operacyjnego komputera (zestaw standardowych funkcji takich jak dostęp do pamięci masowej, komunikacja itp.), najczęściej licencjonowany lub pozyskiwany jako oprogramowanie opensource.

Podsumowując: jeżeli w dalszej treści pojawi się pojęcie "kod źródłowy aplikacji (programu)" mamy na myśli kod komponentu Model.

Pojęcie bezpieczeństwa

Na diagramie Pojęcie bezpieczeństwa pokazano trzy typowe zagrożenia dla komputera. Są to: Zagrożenie ciągłości działania, Zagrożenie dostępu do danych, Zagrożenie ujawnienia mechanizmu działania. W tym miejscu należy przytoczyć definicje pojęcia bezpieczeństwo: słownik języka polskiego PWN definiuje jest jako "stan braku zagrożenia". Pojęcie ryzyka zaś jest definiowane jako "możliwość, że coś się nie uda; też: przedsięwzięcie, którego wynik jest niepewny, które może spowodować szkodę lub stratę".
Jedna z podstawowych reguł współczesnej kryptografii jest Zasada Kerckhoffs'a . Została ona sformułowana pod koniec XIX wieku przez holenderskiego kryptologa Augusta Kerckhoffs'a (Petitcolas, 2011) mówi, że system kryptograficzny powinien być bezpieczny nawet wtedy, gdy są znane wszystkie szczegóły jego działania oprócz sekretnego klucza. Zasada ta może być rozszerzana także na procedury (oprogramowanie), to znaczy, że bezpieczeństwo systemu nie powinno zależeć od faktu ujawnienia procedur utrzymania jego bezpieczeństwa. Zasada ta została przeformułowana (lub prawdopodobnie niezależnie sformułowana) przez amerykańskiego matematyka Claude Shannon jako "wróg zna system" tj. "należy projektować systemy przy założeniu, że wróg natychmiast się z nim w pełni zapozna". Koncepcja jest szeroko akceptowana nie tylko przez przez kryptologów, w przeciwieństwie do zasady "bezpieczeństwo poprzez zaciemnienie" (ukrywanie wiedzy o metodach zabezpieczenia), która jest uznawana za strategię ryzykowną i niepolecaną.



Łańcuch powstawania oprogramowania

Na diagramie Łańcuch powstawania oprogramowania przedstawiono standardowy proces wytwarzania oprogramowania. Są to: opracowanie opisu potrzeb językiem tak zwanym biznesowym (Dziedzinowy opis biznesowy) , opracowanie projektu technicznego (algorytmy, architektura, Opis Techniczny Oprogramowania), na jego podstawie napisanie oprogramowania (Kod programu) oraz umieszczenie w pamięci komputera skompilowanego kodu (Działający kod maszynowy w środowisku komputera) (Object Management Group, 2010). Nie licząc pierwszego etapu, zwanego analizą biznesową, pozostałe dwa to elementy inżynierii oprogramowania. Spotyka się projekty, w których od razu przechodzi się do tworzenia kodu programu jednak w przypadku aplikacji krytycznych pomijanie analizy i projektowania jest uznawane za złą praktykę (Sommerville, 2020).

Kod programu w tym opracowaniu to jedynie kod komponentu Model zobrazowanego na diagramie Wzorzec Architektoniczny MVC, gdyż pozostałe komponenty, także te odpowiedzialne z bezpieczeństwo, to standardowe oprogramowanie pozyskiwane na rynku.
Podsumowując można stwierdzić, że bezpieczeństwo systemu, jakim jest dostępne jego użytkownikom oprogramowanie na określonym komputerze, ma kilka aspektów: aspekt zagrożenia ciągłości działania, aspekt ujawnienia danych, aspekt ujawnienia mechanizmu działania. Zagrożenie ciągłości działania to ryzyko przerwania poprawnej pracy sprzętu (procesor i pamięć komputera) oraz uszkodzenia oprogramowania. Ujawnienie danych wymaga przełamanie procedur ich udostępniania. Ujawnienie mechanizmu działania wymaga dostępu do dokumentacji kodu źródłowego (opis algorytmów i reguł przetwarzania) lub do tego kodu.
Z perspektywy wspomnianej Zasady Kerckhoffs'a, należy uznać, że o jakości i bezpieczeństwie systemu decyduje wyłącznie jakość jego projektu (realizowana logika i procedury). W tym przypadku jest to także jakość środowiska (patrz Controller na diagramie Wzorzec Architektoniczny MVC). Innymi słowy znajomość kodu sama z siebie nie stanowi zagrożenia, zagrożeniem są wady (luki) w realizowanych procedurach (jest to w literaturze przedmiotu podkreślana zaleta oprogramowania open source: o otwartym dostępie do kodu).

4. Opinia dotycząca uzasadnienia decyzji z dnia 4 sierpnia 2022

Minister Sprawiedliwości, decyzją oznaczoną sygnaturą BK-IV.082.270.2022 odmawia udostępnienia kodu źródłowego "Systemu Losowego Przydziału Spraw" słowami:

"*Uprawnione jest bowiem stwierdzenie, że kod źródłowy nie stanowi nośnika informacji o sprawach publicznych. Kod źródłowy to ciąg instrukcji i deklaracji, zapisany w zrozumiałym dla człowieka języku programowania opisujący operacje, jakie powinien wykonać komputer przy pomocy skończonej liczby ściśle zdefiniowanych rozkazów (...) Skoro zatem kod źródłowy jest tylko technicznym elementem przetwarzania danych, a głównym zastosowaniem kodu źródłowego jest wyrażanie programów komputerowych w zrozumiałej postaci należy stwierdzić, że kod źródłowy stanowi jedynie narzędzie wykorzystywane w programach komputerowych, nie zawierając jakiegokolwiek komunikatu o sprawach publicznych (...) kod źródłowy nie stanowi informacji publicznej w rozumieniu art. 1 ust. 1 ustawy o dostępie do informacji publicznej*".
W świetle opisu diagram Wzorzec Architektoniczny MVC oraz diagram Łańcuch powstawania oprogramowania należy jasno stwierdzić, że kod realizujący logikę biznesową czyli funkcjonalność oprogramowania (komponent Model) niesie w 100% informację o tym jakie algorytmy i reguły przetwarzania są przez oprogramowanie realizowane. Kod ten powinien odwzorowywać algorytm opisany w Opisie Technicznym Oprogramowania, a jeżeli taki nie powstał, to w opisie (dokumentacji) algorytmów.
Nie ma żadnych wątpliwości co do tego, że dokument "Dokumentacja analityczna z dnia 14 października, 2021" stanowiąca opis algorytmu losowania składów orzekających dla spraw sądowych, jest informacją publiczną.
W świetle powyższego, kod komponentu Model (a tym samym jego kod źródłowy), który realizuje ten algorytm, stanowi odwzorowanie tego algorytmu a nie, jak twierdzi Minister w Uzasadnieniu:

"*jest tylko technicznym elementem przetwarzania danych, a głównym zastosowaniem kodu źródłowego jest wyrażanie programów komputerowych w zrozumiałej postaci należy stwierdzić, że kod źródłowy stanowi jedynie narzędzie wykorzystywane w programach komputerowych, nie zawierając jakiegokolwiek komunikatu o sprawach publicznych (...)*"

W konsekwencji kod źródłowy komponentu Model, niesie tę samą informację co opis zawarty w Dokumentacji analitycznej, co każe uznać, że skoro są to te same informacje wyrażone w innych językach, to obie formy jej wyrażenia stanowią tę samą informacje, tu sklasyfikowaną już jako publiczną. Nie jest więc prawdą stwierdzenie, że:

"*kod źródłowy nie stanowi informacji publicznej w rozumieniu art. 1 ust. 1 ustawy o dostępie do informacji publicznej*"."

Powyższe potwierdza także polskie orzecznictwo i dyrektywy UE:

"Dokumentacja i inne materiały dotyczące projektowania programu komputerowego należy traktować jako program komputerowy (implementacja dyrektywy Parlamentu Europejskiego I Rady 2009/24/WE z dnia 23 kwietnia 2009 (wersja ujednolicona dyrektywy Rady Wspólnoty Europejskich nr 91/250 z dnia 14 maja 1991): „dla celów niniejszej dyrektywy pojęcie «program komputerowy» obejmuje również przygotowawcze prace projektowe i materiał projektowy".

W dniu 29 Listopada 2011, zapada orzeczenie potwierdzające powyższe:

"Court of Justice of the European Union, PRESS RELEASE No 129/11, Luxembourg, 29 November 2011, Press and Information Advocate General?s Opinion in Case C-406/10, SAS Institute Inc. v World Programming Ltd"

5. Źródła

Nazwa	Opis
(Biernat, 1999)	Biernat, J. (1999). Architektura komputerów. Oficyna Wydawnicza Politechniki Wrocławskiej.
(McMullin, 1985)	McMullin, E. (1985). Galilean idealization. Studies in History and Philosophy of Science Part A, 16(3), 247–273. https://doi.org/10.1016/0039-3681(85)90003-2
(Mrdovic & Perunicic, n.d.)	Mrdovic, S., & Perunicic, B. (n.d.). Kerckhoffs' Principle for Intrusion Detection.
(Murawski, 2015)	Murawski, R. (Ed.). (2015). Filozofia matematyki i informatyki. Copernicus Center Press.
(Object Management Group, 2010)	Object Management Group. (2010). The MDA Foundation Model. Object Management Group. https://www.omg.org/cgi-bin/doc?ormsc/10-09-06.pdf
(Petitcolas, 2011)	Petitcolas, F. A. (2011). Kerckhoffs' Principle.
(Sommerville, 2020)	Sommerville, I., Wlodarz, Marek. (2020). Inzynieria oprogramowania. Wydawnictwo Naukowe PWN.
(Svirca, 2020)	Svirca, Z. (2020, May 30). Everything you need to know about MVC architecture. Medium. https://towardsdatascience.com/everything-you-need-to-know-about-mvc-architecture-3c827930b4c1
(Żeliński, 2020)	Żeliński, J. (2020). Systematyzacja pojęć analizy i projektowania obiektowego. 18. https://doi.org/10.13140/RG.2.2.18216.52489

6. Jarosław Żeliński.

Absolwent Wojskowej Akademii Technicznej. W latach 1989–1991 adiunkt. Od 1991 roku jako specjalista w obszarze projektowania systemów informatycznych dużej skali. W latach 2000-2004 cztery roczne kontakty dla największych firm telekomunikacyjnych jako analityk i projektant systemów. Do odbiorców usług należą także między innymi: KGHM Polska Miedź SA, Kancelaria Senatu, Komisja Nadzoru Finansowego, centralne instytucje publiczne, zakłady produkcyjne i usługowe. Od 1998 prowadzi samodzielne prace badawcze nad stosowaniem modeli w analizach systemowych. Od 2004 roku prowadzi niezależną działalność zawodową jako ekspert w obszarze analiz systemowych i projektowania systemów. Prowadzi wykłady z zakresu analizy biznesowej i modelowania, na uczelniach wyższych: początkowo od 2005 roku Akademia Morska w Gdyni, od 2015 roku w WIT Wyższa Szkoła Informatyki Stosowanej i Zarządzania pod auspicjami Polskiej Akademii Nauk. W dorobku kilkadziesiąt publikacji w prasie branżowej, literaturze naukowej, w tym: w 2016 roku, nakładem wydawnictwa One Press, ukazała się autorska monografia Analiza Biznesowa, w roku 2017 i 2020 wysoko ocenione, recenzowane publikacje naukowe w USA (wydawca: IGI Global) w obszarze zarządzania informacją (ORCID 0000–0002-8032–4720).

Jarosław Żeliński, +48 608 05 90 20, j.zelinski@it-consulting.pl, https://IT-Consulting.pl