Bezpieczeństwo w inżynierii oprogramowania

Wprowadzenie

Arkadiusz Chrobot

Katedra Systemów Informatycznych

5 października 2024

Plan

Informacje kontaktowe

Wykładowca: dr inż. Arkadiusz Chrobot

Numer pokoju: 3.23, budynek D

Termin konsultacji: czwartek, 12:00 – 14:00

Numer telefonu: 41 34-24-185

Adres e-mail: a.chrobot@tu.kielce.pl

Strona WWW: https://achilles.tu.kielce.pl/portal/Members/84df831b59534bdc88bef09b15e73c99

Informacje organizacyjne

▸ Karta przedmiotu

Tematyka wykładów

- Secure Development Life Cycle
- Mechanizmy uwierzytelniania
- Architektura i bezpieczeństwo
- Wzorce bezpieczeństwa
- Metody formalne
- Testowanie jednostkowe i integracyjne, a bezpieczeństwo
- Bezpieczeństwo w fazie utrzymania i wycofania systemu
- DevSecOps


Ross Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley, 2020. URL: https://www.cl.cam.ac.uk/~rja14/book.html.


Loren Kohnfelder. *Po pierwsze: bezpieczeństwo, Przewodnik dla twórców oprogramowania*. Gliwice: Helion, 2022.


Bass Len, Clements Paul i Kazman Rick. *Architektura oprogramowania w praktyce*. Wydanie IV. Gliwice: Helion, 2022.


David A. Wheeler. *Developing Secure Software (LFD121)*. 2024. URL: https://training.linuxfoundation.org/training/developing-secure-software-lfd121/.


Safecode Training. 2023. URL: https://safecode.org/training/.


Security Development Lifecycle. 2024. URL: https://www.microsoft.com/en-us/download/details.aspx?id=16420.

Podstawowe pojęcia

Definicja IEEE 610.2 (tłumaczenie własne)

„Inżynieria oprogramowania jest zastosowaniem systematycznego, zdyscyplinowanego i mierzalnego podejścia do wytwarzania, wdrażania i utrzymywania (ang. *maintenance*) oprogramowania; tzn. zastosowaniem inżynierii do oprogramowania i badaniem takiego podejścia."

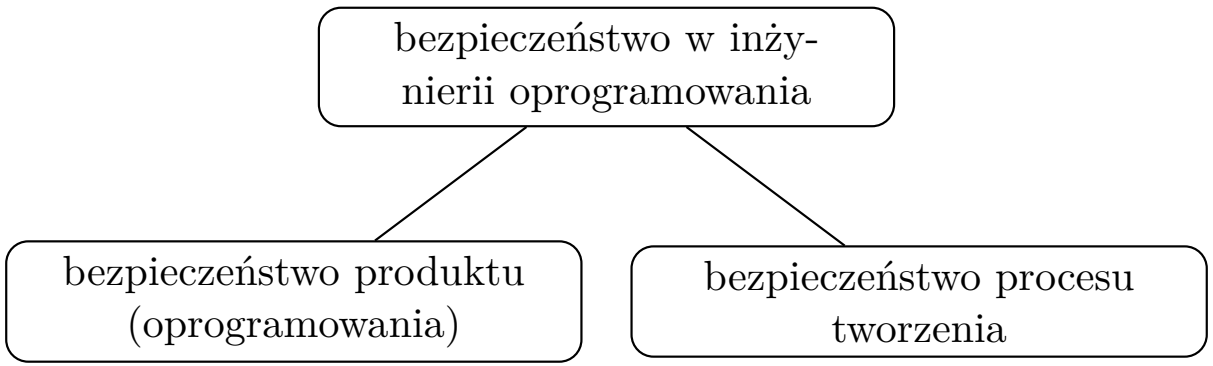
Tradycyjne pojmowanie bezpieczeństwa w inżynierii oprogramowania

Bezpieczeństwo jest *niefunkcjonalnym* (*pozafunkcjonalnym*) wymaganiem, które musi być spełnione przez oprogramowanie. Jest to także wymaganie *złożone*.

Podstawowe pojęcia

Ale czym ogólnie jest bezpieczeństwo (cyberbezpieczeństwo)? To pojęcie, które ▸ ewoluuje:
całkowicie zabezpieczone systemy (ang. *secure systems*) → **zaufane systemy** (ang. *trusted systems*) → **godne zaufania systemy** → **odporne systemy** (ang. *resilient*) → **systemy o zredukowanym ryzyku** (ang. *risk-reduction systems*)
Podobnie ewoluuje postrzeganie bezpieczeństwa w inżynierii oprogramowania. Współcześnie nie tylko uważa się, że ma wpływ na większą ▸ liczbę kategorii wymagań (funkcjonalne, niefunkcjonalne, wynikowe (ang. *derived*)), ale również, że dotyczy większej liczby aspektów inżynierii oprogramowania.

Podstawowe pojęcia



Związek bezpieczeństwa z inżynierią oprogramowania

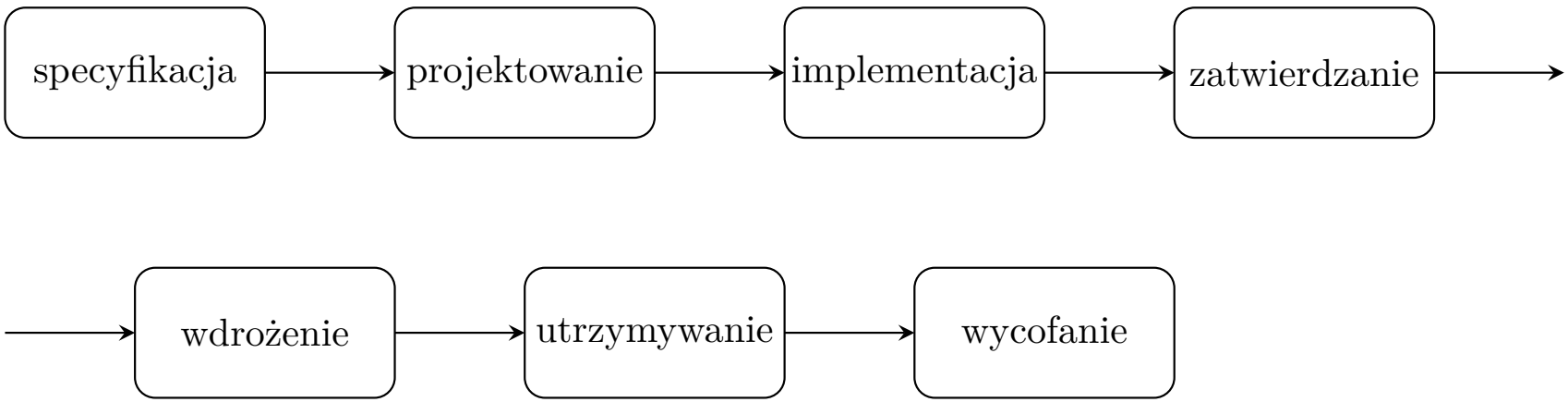
Konsekwencje zaniedbań

Dlaczego bezpieczeństwo procesu tworzenia i produktu jest ważne?

- Solarwinds (2020) — przełamanie zabezpieczeń tego dostawcy usług IT pozwoliło intruzom umieścić tyle furtki w oprogramowaniu, które następnie zostało zainstalowane w agencjach rządowych USA oraz przez jedne z największych korporacji.
- MOVEit (2023) — intruzi odkryli i wykorzystali lukę w oprogramowaniu do przesyłania plików MOVEit, pozwalającą im wykraść dane z korporacji zaliczanych do Big4, z UCLA i Siemens Energy.
- Barracuda Networks (2023) — intruzi uzyskali dostęp do danych użytkowników wykorzystując podatność w systemie obsługi poczty elektronicznej.
- XZ (2024) — intruz ▸ przejął kontrolę nad repozytorium biblioteki kompresji XZ, aby umieścić tylną furtkę w SSH.

▸ Źródło informacji trzech pierwszych incydentach.

Cykl życia oprogramowania



Ogólny cykl życia oprogramowania

Wymagania Secure Development Life Cycle

- polityki i procedury,
- role i odpowiedzialności,
- wytyczne,
- specjalizowane zasoby (np. narzędzia).

Charakterystyka Secure Development Life Cycle

Secure Development Life Cycle (SDL) zawiera te same fazy, co większość procesów wytwórczych oprogramowania, ale także dodatkowe, związane z cyberbezpieczeństwem. Jego procedury i polityki są całkowicie zintegrowane z procesem tworzenia. SDL może być dostosowywany do różnych do różnych metod opracowywania oprogramowania i rozszerzany. Aby ułatwić jego wdrożenie, zawiera zazwyczaj model rozwojowy.

Przykłady Secure Development Life Cycle

Historycznie pierwszym opracowanym jest SDL firmy Microsoft. Zasady (SD3+C):

1. zaprojektowane by być bezpieczne (ang. *secure by design*) — bezpieczna architektura i struktura, modelowanie zagrożeń, eliminacja podatności właściwych dla dziedziny zastosowania;
2. bezpieczne do wdrożenia (ang. *secure in deployment*) — wytyczne wdrożenia, zastosowanie narzędzi analizy i zarządzania do ustalenia najlepszej pod względem bezpieczeństwa konfiguracji, narzędzia dystrybucji poprawek bezpieczeństwa;
3. domyślnie bezpieczne (ang. *secure by default*) — zasada najmniejszego uprzywilejowania, zasada głębokiej obrony, zachowawcze ustawienia domyślne, domyślnie wyłączone rzadziej stosowane usługi;
4. komunikacja (ang. *communication*) — kontakt z użytkownikiem, zasady reagowania na incydenty bezpieczeństwa.

Przykłady Secure Development Life Cycle

SDL firmy Microsoft bazuje na klasycznych metoda tworzenia oprogramowania. Struktura tego procesu:

1. Szkolenia w modelowaniu zagrożeń, programowaniu zabezpieczeń, testowaniu bezpieczeństwa, projektowaniu zabezpieczeń, zapewnianiu prywatności.
2. Ustalenie wymagań bezpieczeństwa, opracowanie bramek jakości, limitów defektów, oszacowanie ryzyk dla bezpieczeństwa i prywatności (**specyfikacja**).
3. Ustalenie wymagań projektowych, analiza powierzchni ataku, modelowanie zagrożeń (**projektowanie**).
4. Użycie zatwierdzonych narzędzi, wycofanie niebezpiecznych funkcji (API), analiza statyczne (**implementacja**).
5. Dynamiczna analiza, testowanie rozmyte, przegląd powierzchni ataku (**zatwierdzenie/weryfikacja**).
6. Plany reakcji na incydenty, ostateczny przegląd bezpieczeństwa, archiwizacja wersji (**wdrożenie**).

Przykłady Secure Development Life Cycle

OWASP opracowała własny model SDL, dla klasycznego podejścia do tworzenia oprogramowania, który składa się z pięciu faz. Paradygmat ten nazywa się OWASP SAMM.

1. **Zarządzanie** — opracowanie i wdrożenie strategii i metryk, wprowadzenie polityk i standardów, szkolenia.
2. **Projektowanie** — ustalenie zagrożeń, określenie wymagań, bezpieczna architektura.
3. **Implementacja** — bezpieczne tworzenie, bezpieczne wdrożenie, zarządzanie defektami.
4. **Weryfikacja** — ocena architektury, testowanie sterowane wymaganiami, testy bezpieczeństwa.
5. **Utrzymanie** — zrządzanie incydentami, zarządzanie środowiskami, zarządzanie operacyjne.

Secure Development Life Cycle dla metod zwinnych

SDL-Agile:

- Nie każda praktyka SDL i każde wymaganie SDL musi być spełnione w każdym sprincie.
- Każdy zespół musi wnieś wystarczającą ilość pracy związanej z SDL implementacją bieżącego udogodnienia.
- SDL-Agile definiuje zadania, które mogą być odwzorowane w zwinnym procesie rozwoju oprogramowania.
- Jeśli jest to uzasadnione i bezpieczne, zespół może pominąć w określonym sprincie pewne czynności SDL.
- Wymagania SDL są podzielone na trzy kategorie: dla sprintu, kubełkowe (ang. *bucket*) — weryfikacja, przegląd projektu, planowanie — i jednorazowe.

Pytania

?

Koniec

Dziękuję Państwu za uwagę!