Bezpieczeństwo w inżynierii oprogramowania

DevSecOps

Arkadiusz Chrobot, Adam Malicki

Katedra Systemów Informatycznych

14 grudnia 2024

Plan

1. Wstęp
2. DevOps
3. DevSecOps

Wstęp

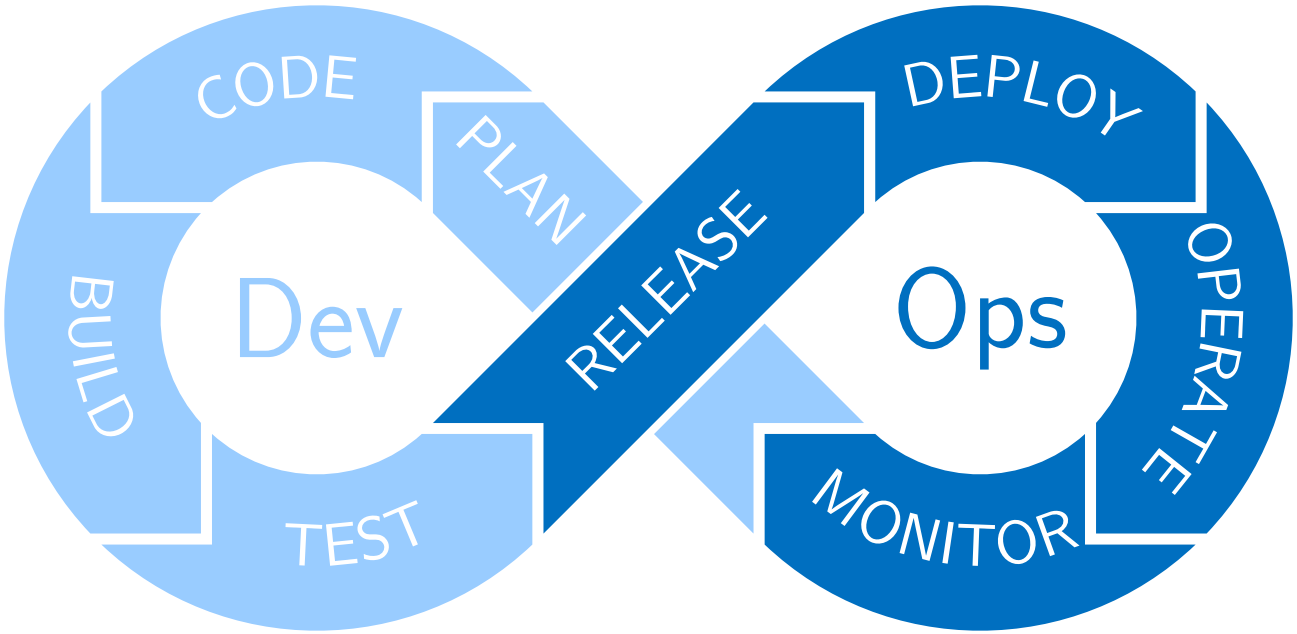
Jednym z nadrzędnych celów inżynierii oprogramowania jest zapewnienie wysokiej jakości produktu. Do jej zapewniania potrzebna jest obszerna informacja zwrotna uzyskiwana na drodze testów. Dlatego pojawił się paradygmat *przesunięcia w lewo* (ang. *shift left*) testów. Termin ten oznacza wdrożenie testowania już w najwcześniejszych fazach projektu związanego z tworzeniem i utrzymaniem oprogramowania.

To podejście jest stosowane zarówno w tradycyjnych, jak i zwinnych metodach wytwarzania oprogramowania, ale nabrało szczególnego znaczenia, kiedy pojawiły się technologie chmurowe i rozpowszechniły związane z nimi rozwiązania, takie jak maszyny wirtualne i kontenery. Tworzenie oprogramowania dla takich środowisk wymaga połączenia prac rozwojowych (ang. *development*) z operacyjnymi (ang. *operations*). Stąd powstał model DevOps. Ponieważ w systemach chmurowych ważnym zagadnieniem jest bezpieczeństwo, to wkrótce na bazie DevOps opracowano paradygmat DevSecOps.

Wprowadzenie do DevOps

Obecnie najczęściej stosowanym podejście do wytwarzania i utrzymania oprogramowania są *metody zwinne* (ang. *agile*), w których oprogramowanie jest tworzone przyrostowo i iteracyjnie, co oznacza częste jego aktualizacje i wdrożenia. Do tej charakterystyki pasują bardzo dobrze środowiska chmurowe, pozwalające na zastosowanie w oprogramowaniu architektury mikrousług (ang. *microservices*) poprzez wykorzystanie maszyn wirtualnych i (głównie) kontenerów. Ta zmiana techniczna spowodowała powstanie nowego podejścia do rozwoju oprogramowania nazwanego DevOps [1, 2]. Proces ten składa się z kilku faz (Rysunek 1) związanych z tworzeniem oprogramowania (Dev: planowanie, implementowanie, kompilacja, testy) i zapewnieniem jego działania (Ops: wdrożenie, utrzymanie, monitorowanie), które wykonywane są często i relatywnie szybko. Dlatego ich efektywność zależy od stopnia zautomatyzowania czynności z nimi związanych.

Wprowadzenie do DevOps



Model DevOps (źródło: https://tex.stackexchange.com/questions/723089/create-devops-cycle-in-tikz)

Fazy DevOps

Proces DevOps [1] obejmuje następujące fazy.

planowanie (ang. *plan*) w tej fazie wykonywane są czynności, które pozwalają zarządzać czasem, kosztami i jakością, takie jak analiza wykonalności, określanie wymagań, projektowanie;

implementowanie (ang. coding) przyrostowe tworzenie kodu oprogramowania;

kompilacja (ang. *build*) integrowanie i kompilowanie przyrostów opracowanych przez wielu programistów;

testowanie (ang. *test*) wykonywanie testów statycznych i dynamicznych na kilku poziomach (integracyjne, systemowe, akceptacyjne);

wydanie (ang. *release*) przygotowanie i wydanie wersji do instalacji;

wdrożenie (ang. *deploy*) instalacja wersji oprogramowania lub jej zaktualizowanych komponentów;

Fazy DevOps

utrzymanie (ang. *operate*) skalowanie, równoważenie obciążenia, tworzenie kopii zapasowych.

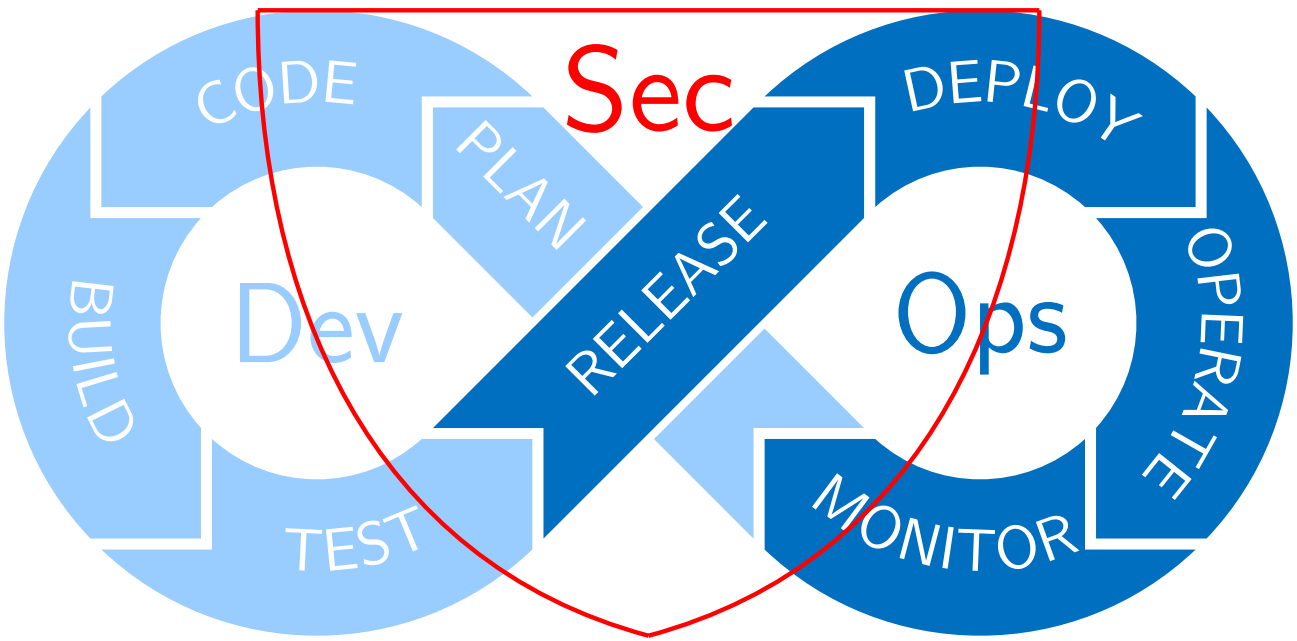
minitorowanie (ang. *monitoring*) zbieranie metryk dotyczących działania oprogramowania, analiza.

Efektywność wykonywania tych faz wymaga zautomatyzowania jak największej liczby czynności związanych z DevOps. Od strony technicznej najważniejszym elementem *ekosystemu* DevOps jest potok CI/CD (ang. *Continuous Integration/Continuous Delivery*), czyli *ciągłej integracji i ciągłego dostarczania*. Pozwala on nie tylko na integrację przyrostów kodu tworzonego przez programistów, ale również na jego kompilację, weryfikowanie, instalację i monitorowanie. Wszystkie te czynności są w jak największym stopniu zautomatyzowane. Firma tworząca oprogramowanie może mieć skonfigurowanych kilka potoków CI/CD, każdy dostosowany do potrzeb opracowywanego oprogramowania. Razem tworzą one *fabrykę oprogramowania* (ang. *software factory*).

Wprowadzenie do DevSecOps

Paradygmat DevOps, tak jak inne modele tworzenia oprogramowania, ma na celu zwiększenie jakości produktu. W środowiskach chmurowych jednym z najistotniejszych jej składników jest bezpieczeństwo. Z tego powodu uzupełniono DevOps o działania zmierzające do jego zapewnienia (Rysunek 2). W ten sposób powstał paradygmat DevSecOps, w którym połączono prace rozwojowe, z operacyjnymi i dotyczącymi bezpieczeństwa. Głównym jego założeniem jest „przesunięcie w lewo" nie tylko testów, ale również kwestii związanych z bezpieczeństwem. Za jego utrzymanie stają się odpowiedzialne nie tylko osoby na stanowiskach bezpośrednio związanych z zabezpieczeniami, ale również wszyscy inżynierowie oprogramowania i pracownicy IT.

Wprowadzenie do DevSecOps

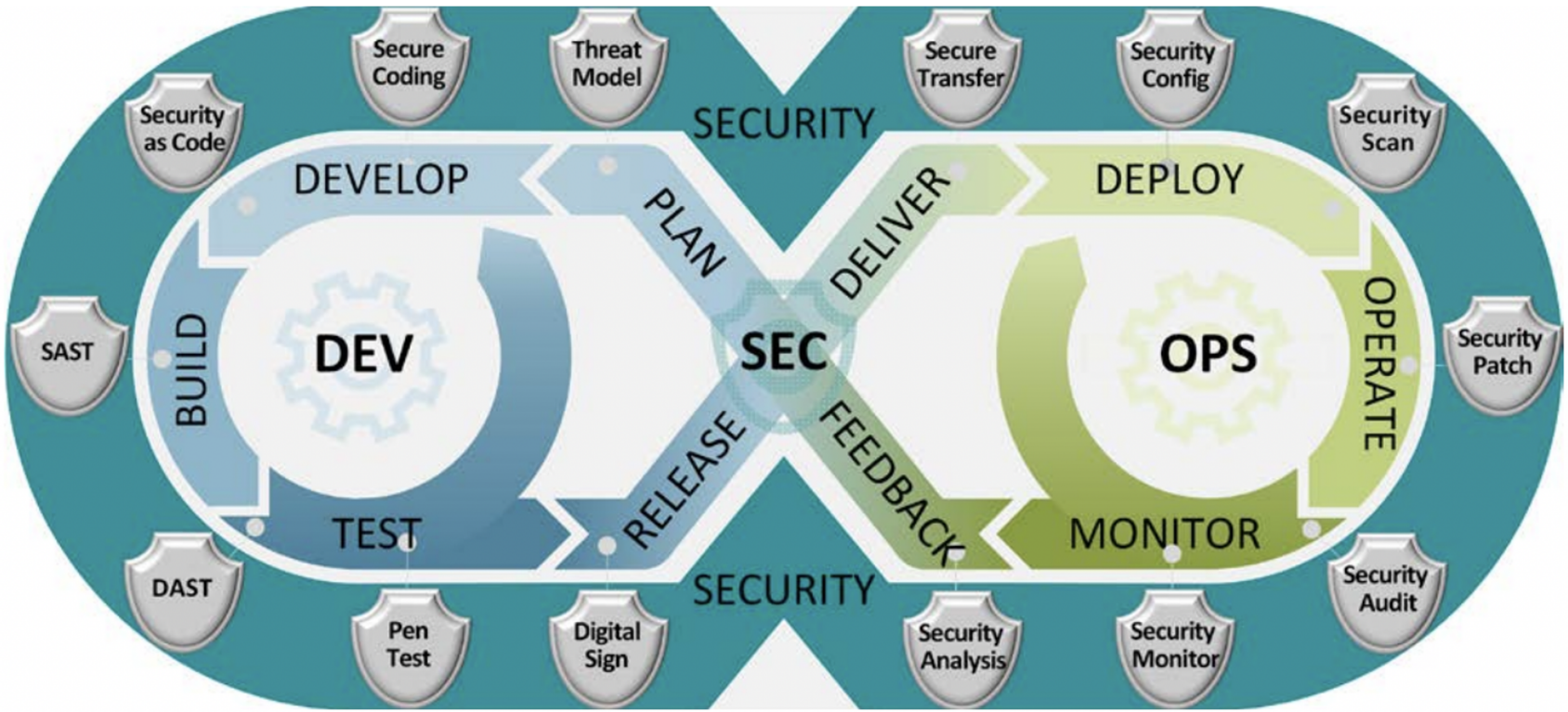


Model DevSecOps (na podstawie: https://tex.stackexchange.com/questions/723089/create-devops-cycle-in-tikz)

Wprowadzenie do DevSecOps

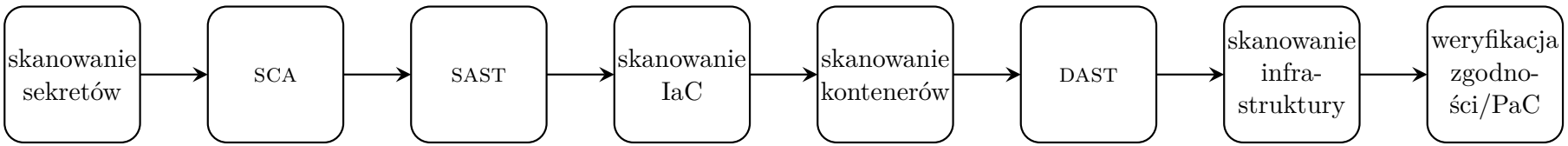
W DevSecOps zagadnienia związane z bezpieczeństwem są uwzględniane w każdej fazie procesu tworzenia i utrzymania oprogramowania (Rysunek 3). Działania z nimi związane powinny być w największym możliwym stopniu zautomatyzowane, lecz nie bardziej, ani mniej. Oznacza to, że czynności wymagające udziału człowieka, takie jak np. testowanie statyczne, polegające na ręcznym przeglądzie kodu, powinny być przez narzędzia tylko wspierane. Natomiast te, które tego nie wymagają, jak np. konfigurowanie zabezpieczeń, mogą odbywać się całkowicie automatycznie. Dlatego w DevSecOps oprócz koncepcji *infrastruktura jako kod* (ang. *Infrastructure as Code*) stosowane jest podejście *bezpieczeństwo jako kod* (ang. *Security as Code*).

Wprowadzenie do DevSecOps



Elementy związane z bezpieczeństwem w DevSecOps (źródło: [1, 3])

Potok DevSecOps



Potok DevSecOps (na podstawie [4])

Potok CI/CD w DevSecOps, podobnie jak w DevOps może być tworzony przyrostowo [1]. Do typowych elementów potoku CI/CD zalicza się narzędzie sterujące pracą tego potoku, takie jak na przykład ▸ Jenkins, ▸ GitLab, ▸ TeamCity, repozytorium artefaktów, zawierające biblioteki i inne komponenty (przykład: ▸ Artifactory) oraz środowiska rozwojowe (ang. *development environment*), testowe (ang. *test environment*), przedprodukcyjne (ang. *pre-production environment*) i produkcyjne (ang. *production environment*) oraz repozytorium kodu (typowo: git).

Potok DevSecOps

Model DevSecOps wymaga wprowadzenia do potoku CI/CD dodatkowych narzędzi (Rysunek 4), takich jak:

Skaner sekretów (ang. *secret scanner*) narzędzie, które poszukuje wpisanych na stałe do oprogramowania haseł i innych informacji uwierzytelniających. Skanowanie to powinno dotyczyć nowego kodu, *przed* dodaniem go do repozytorium.

SCA (*Software Composition Analyzer*) narzędzie, które sprawdza, czy w trakcie kompilacji dodawane są do oprogramowania gotowe komponenty niemające znanych podatności.

SAST (*Static Application Security Testing*) narzędzie, które sprawdza kod oprogramowania, bez jego uruchamiania, czy nie zawiera on defektów będących podatnościami,

Skaner IaC (ang. *Infrastructure as Code*) oprogramowanie, które sprawdza, czy w konfiguracji infrastruktury nie ma luk bezpieczeństwa.

Potok DevSecOps

Skaner kontenerów narzędzie, które weryfikuje stan zabezpieczeń i aktualność kontenerów.

DAST (*Dynamic Application Security Testing*) narzędzia testujące mechanizmy bezpieczeństwa oprogramowania w trakcie jego wykonania.

Skaner infrastruktury narzędzia monitorujące stan infrastruktury, czyli np. zużycie zasobów.

weryfikatory zgodności narzędzia sprawdzające, czy parametry oprogramowania odpowiadają umowie z klientem i innym regulacjom oraz standardom.

DevSecOps — Planowanie

W fazie planowania, oprócz czynności typowych dla DevOps, w DevSecOps dokonuje się modelowania zagrożeń i oceny ryzyk. Do zespołu są dodawane osoby będące ekspertami w zakresie bezpieczeństwa (ang. *Subject/Security Matter Expert*). Ponadto wprowadza się szkolenia zarówno dla inżynierów oprogramowania jak i personelu IT. W tej fazie stosuje się narzędzia umożliwiające komunikację między członkami zespołu (np. oprogramowanie do telekonferencji), ale również specjalistyczne programy, np. do modelowania zagrożeń, jak opracowane przez OWASP narzędzie ▸ Threat Dragon.

DevSecOps — Planowanie

Przykładowe szkolenie dla programistów może obejmować zasady bezpiecznego kodu proponowane przez OWASP [4]:

1. zaimplementuj autoryzację,
2. użyj kryptografii do ochrony danych,
3. stosuj walidację danych wejściowych i obsługę wyjątków,
4. stosuj bezpieczeństwo od samego początku,
5. twórz konfiguracje domyślnie bezpieczne,
6. utrzymuj bezpieczeństwo komponentów,
7. stosuj uwierzytelnianie (bezpieczne cyfrowe tożsamości),
8. użyj mechanizmów bezpieczeństwa wbudowanych w przeglądarki,
9. zaimplementuj logowanie i monitorowanie zdarzeń związanych z bezpieczeństwem,
10. powstrzymaj fałszowanie żądań po stronie serwera.

DevSecOps — implementowanie i kompilowanie

W fazach implementacji (ang. *develop*) i kompilacji (ang. *build*) oprócz zintegrowanych środowisk programistycznych (ang. *Integrated Development Environment* — IDE), repozytoriów kodu i artefaktów oraz kompilatorów, stosuje się wtyczki (ang. *plugins*) do IDE, które dokonują bieżącej analizy kodu pod względem bezpieczeństwa. Ponadto do potoku CI/CD dodawane są skanery sekretów, sprawdzające kod *przed* dodaniem go do repozytorium, narzędzia typu SAST do statycznej analizy kodu pod kątem bezpieczeństwa (taką rolę może np. dogrywać ▸ SonarQube). W fazie implementacji są również prowadzone przeglądy kodu, także z uwzględnieniem zagadnień dotyczących bezpieczeństwa. Przygotowywana jest konfiguracja zabezpieczeń na drodze programowej (ang. *Software as Code*). Zarówno repozytorium kodu, jak i repozytorium artefaktów powinny być zabezpieczone przed nieautoryzowanym dostępem. Dodatkowo powinny one umożliwiać separację kodu poszczególnych projektów, jeśli obsługują ich wiele.

DevSecOps — Testowanie

Faza testowania dotyczy głównie testów dynamicznych, mających na celu sprawdzanie spełnienia przez tworzone oprogramowania wymagań niefunkcjonalnych. Jednym z nich jest bezpieczeństwo, które może być weryfikowane przy pomocy narzędzi typu DAST, które badają zachowanie oprogramowania w trakcie jego działania, w poszukiwaniu problemów związanych z konfiguracją środowiska, uwierzytelnianiem lub walidacją danych wejściowych. Te narzędzia mogą wykonywać testy automatycznie lub być użyte w przeprowadzanych manualnie *testach penetracyjnych*. Zalicza się do nich ▸ OWASP ZAP i ▸ Burp Suite. Innym typem narzędzi stosowanym w DevSecOps są narzędzia IAST (ang. *Interactive Application Security Testing*). Monitorują one bezpieczeństwo aplikacji w trakcie prowadzonych testów, nawet jeśli nie są one związane bezpośrednio z weryfikacją bezpieczeństwa i raportują o wykrytych nieprawidłowościach.

DevSecOps — Wydawanie wersji

Przed instalacją na środowisku produkcyjnym wersja oprogramowania lub jego przyrost musi być także sprawdzony na środowisku przedprodukcyjnym, którego konfiguracja powinna być w jak największym stopniu zbliżona do środowiska produkcyjnego. Weryfikacja te może uwzględniać np. sprawdzenie, czy oprogramowanie może działać bezpiecznie w środowisku zastępczym, gdyby np. podstawowe z jakiś powodów nie było dostępne. Podjęcie decyzji o wydaniu wersji jest dokonywane automatycznie po zweryfikowaniu jej poziomu jakości. Wersja przed dostarczeniem do środowiska produkcyjnego jest podpisywana cyfrowo. Kontenery zawierające tę wersję powinny być także sprawdzone pod kątem zabezpieczeń i aktualności.

DevSecOps — Instalacja

Instalacja rozpoczyna się od weryfikacji podpisu cyfrowego. Powinna polegać na *podmianie* (ang. *replacement*), a nie modyfikacji kontenerów lub maszyn wirtualnych. Ich obrazy powinny być niezmienne, aby zapewnić odpowiedni poziom bezpieczeństwa.

DevSecOps — Utrzymanie i monitorowanie

W tych fazach monitorowaniu podlegają kluczowe wskaźniki wydajności (ang. *Key Performance Indicators* — KPI) oprogramowania oraz parametry infrastruktury, które świadczą o poziomie jej bezpieczeństwa. Do typowych metryk rejestrowanych w DevSecOps należą [3]:

- częstotliwość instalacji,
- czas od dodania do repozytorium do instalacji,
- średni czas naprawy (ang. *Mean Time To Recovery — MTTR*) lub średni czas między awariami (ang. *Mean Time Between Failures MTBF*),
- zmiany w częstości awarii.

Do przetwarzania wspomnianych danych mogą służyć narzędzia takie jak ▸ Grafana i ▸ Prometheus. Pierwsze z nich umożliwia łączenie metryk, dzienników, rejestrów i wizualizację oraz analizę danych telemetrycznych w czasie rzeczywistym. Drugie służy m.in. do akwizycji danych i może stanowić ich źródło dla Grafany.

DevSecOps — Utrzymanie i monitorowanie

Wymienione na poprzednim slajdzie metryki mogą nie być wystarczające w przypadku DevSecOps i powodować powstanie długu technicznego w oprogramowaniu. Proponowane jest stosowanie dodatkowych pomiarów:

wolumen zmian liczba dostarczonych *user stories* w określonym czasie;

dostępność liczba przestojów/wznowień działania w danym okresie;

wolumen problemów klienta liczba zgłoszonych przez klienta problemów w określonym czasie;

liczba rozwiązanych problemów klienta liczba zgłoszonych przez klienta problemów, które udało się rozwiązać;

czas dostarczenia wartości czas od zgłoszenia zapotrzebowania na usługę w oprogramowaniu, do jej dostarczenia;

DevSecOps — Utrzymanie i monitorowanie

czas do uzyskania pozwolenia na wydanie czas od chwili rozpoczęcia projektu do uzyskania wersji nadającej się do instalacji;

czas likwidowania luk czas od wykrycia podatności do wydania poprawki.

Niestety, nie wszystkie z proponowanych metryk są mierzalne w automatyczny sposób oraz nie wszystkie spełniają kryteria dobrych metryk.

DevSecOps — Ocena dojrzałości

OWASP opracował wytyczne [5] pozwalające ocenić dojrzałość procesu DevSecOps stosowanego w danej firmie i stosowanych w ramach niego rozwiązań. Powstało również narzędzie ułatwiające przeprowadzenie takiej oceny, w postaci strony WWW.

Źródła I


DoD Enterprise DevSecOps Reference Design. URL: https://dodcio.defense.gov/Portals/0/Documents/DoD%20Enterprise%20DevSecOps%20Reference%20Design%20v1.0_Public%20Release.pdf (termin wizyty 09. 12. 2024).


DevSecOps. URL: https://www.hostersi.pl/oferta/devsecops/ (termin wizyty 09. 12. 2024).


Bill Nichols. The Current State of DevSecOps Metrics. Carnegie Mellon University, Software Engineering Institute's Insights (blog), marzec 2021. URL: https://insights.sei.cmu.edu/blog/the-current-state-of-devsecops-metrics/. Accessed: 2024-Dec-7.


OWASP DevSecOps Guideline. URL: https://github.com/OWASP/DevSecOpsGuideline/tree/master?tab=readme-ov-file (termin wizyty 09. 12. 2024).

Źródła II


OWASP Devsecops Maturity Model. URL: https://owasp.org/www-project-devsecops-maturity-model/ (termin wizyty 09.12.2024).

Pytania

?

Koniec

Dziękuję Państwu za uwagę!