Bezpieczeństwo aplikacyjne

Aleksander P. Czarnowski

AVET Information and Network Security Sp. z o.o.

AVET
Information and Network Security

Agenda

- Kilka słów o SDL
- Przypadki
- Cloud a bezpieczeństwo aplikacyjne
- Podsumowanie

AVET
Information and Network Security

Jak wytwarzać bezpieczne aplikacje

SECURE DEVELOPMENT LIFECYCLE

AVET
Information and Network Security

Podstawowe elementy składowe bezpiecznej aplikacji

Założenia biznesowe

Projekt

Architektura

Bezpieczne programowanie	Testowanie	Środowiska wykonania
Standardy i procedury	Scenariusze testów	Procedury eksploatacyjne

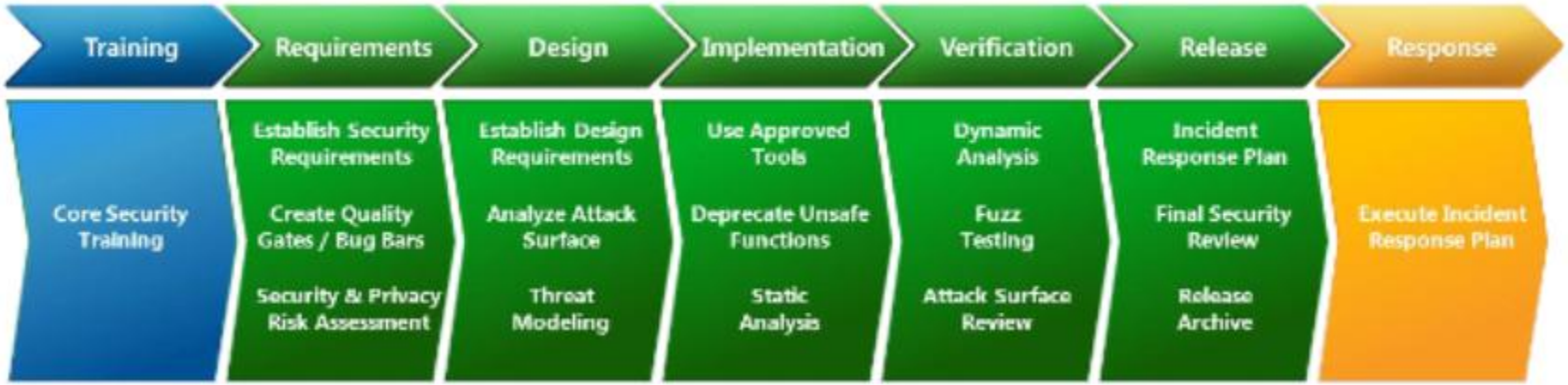
AVET
Information and Network Security

SDL

- Secure Development Lifecycle (SDL/SDLC)
- Doświadczenia Microsoft z ostatnich 10 lat prowadzenia programu
- Różnice pomiędzy programami prowadzonymi w Polsce a na świecie
- Zastosowanie SDL w praktyce

AVET
Information and Network Security

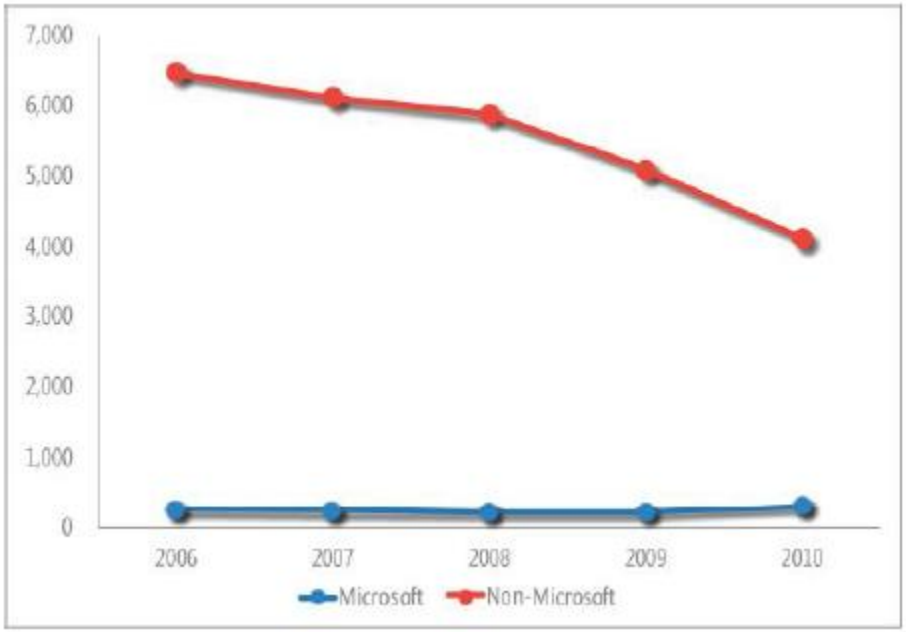
Secure Development Lifecycle

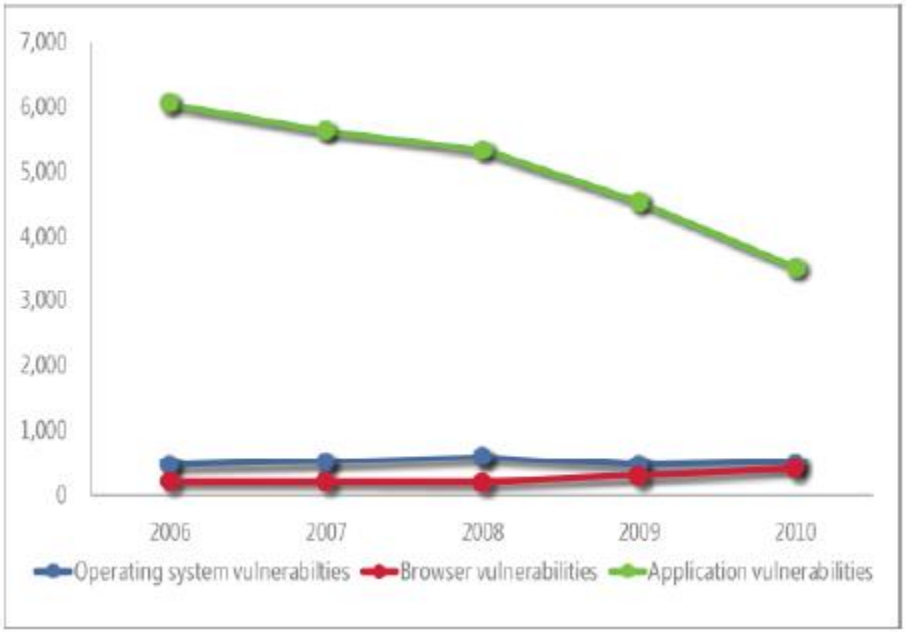


AVET
Information and Network Security

SDL – czy warto?

- Przypadek Microsoft





AVET
Information and Network Security

Różnice w realizacji SDL

Polska

- Bezpieczeństwo staje się ważną funkcją lub wymaganiem
- Jakość nie jest jeszcze postrzegana jako problem
- Zastępy testerów nie nadążają z testami – tymczasem jakość oprogramowania nie rośnie wraz ze wzrostem liczby testów
- Relatywnie niski budżet na SDL
- Brak stałego zespołu programistów

USA/Europa

- Bezpieczeństwo jest istotną funkcjonalnością
- SDL może mieć odpowiedni budżet i zespół
- Silny outsourcing, często w Indiach = istotne różnice kulturowe
- Stałe zespoły programistyczne

Modelowanie zagrożeń

- Usystematyzowana metoda identyfikacji zagrożeń charakterystycznych dla danego systemu
- Aby zapewnić rzeczywistą wartość dodaną proces musi spełniać minimum następujące warunki:
  - Słowniki: pojęć, zagrożeń, podatności i komponentów
  - Powtarzalność
  - Efektywność czasową

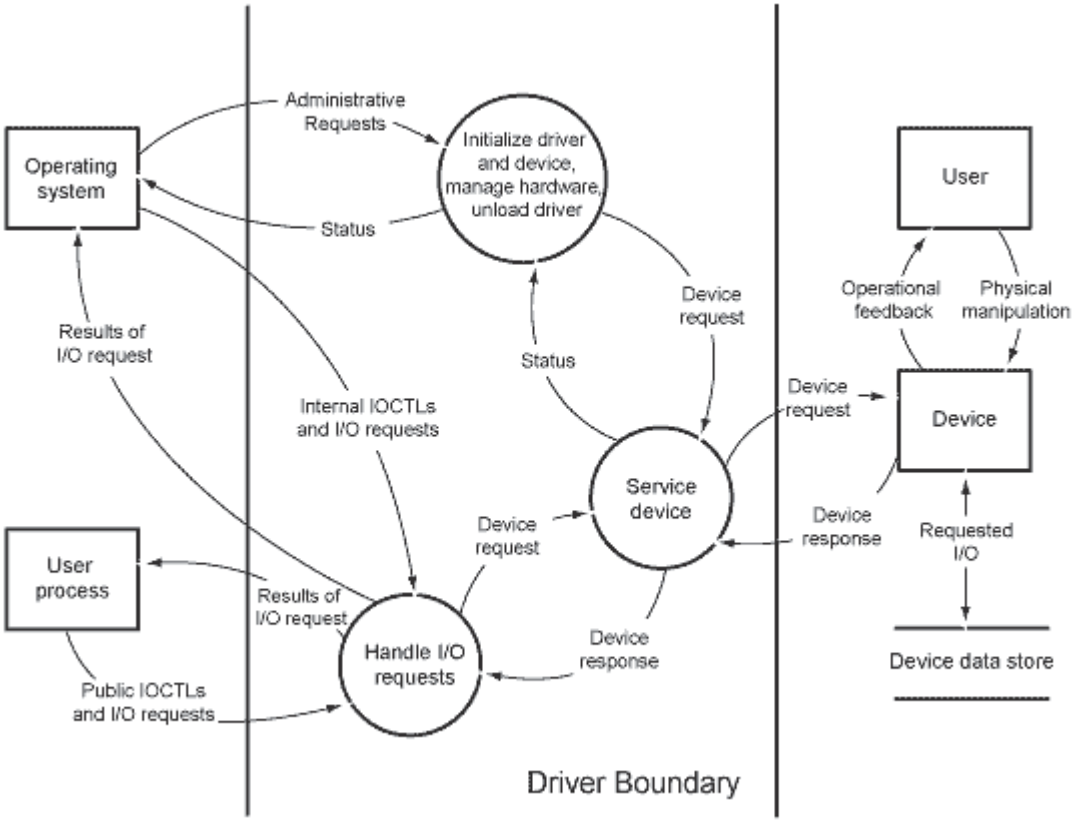
AVET
Information and Network Security

Modelowanie zagrożeń

- Identyfikacja problemów w obszarze projektu
- Identyfikacja problemów w obszarze architektury
- Identyfikacja konfliktów w procesie uwierzytelnienia
- Identyfikacja konfliktów w procesie zarządzania bezpieczeństwem
- Identyfikacja minimalnych wymaganych uprawnień (zazwyczaj jednak bardzo wysokopoziomowo)
- Zrozumienie i nadanie priorytetów zidentyfikowanemu ryzyku

AVET
Information and Network Security

Przykład



Źródło: http://msdn.microsoft.com/en-us/windows/hardware/gg487497

Threat Modeling for Drivers

Przypadek #1: ochrona aplikacji w środowisku produkcyjnym

USUWAJ PODATNOŚCI U ICH ŹRÓDŁA

AVET
Information and Network Security

Podstawowe założenia

- Aplikacja webowa
  - Projekt dotyczy istniejącej aplikacji
  - Aplikacja działa już na środowisku produkcyjnym
  - Brak dobrego i zgranego zespołu programistycznego
  - Nieaktualne wersje w repozytorium
  - Brak dokumentacji projektowej
  - Fatalne udokumentowanie kodu
  - Około 2 tygodni kalendarzowych na uporządkowanie sytuacji

AVET
Information and Network Security

Wyniki wstępnego audytu

- Bałagan w repozytorium kodu = bałagan w kolejnych releasach = problem rollback'u
- Możliwość dostępu programistów do środowiska produkcyjnego z prawami administratora
  - Edycja kodu w ramach poprawek w trybie „online"
  - Nikt nie wie, która tak naprawdę wersja oprogramowania działa w środowisku produkcyjnym
- Brak systemu śledzenia zgłoszeń
  - Brak procedury zarządzania poprawkami
- Modelowanie zagrożeń:
  - Nie ma na to czasu (nawet w przypadku metody rapid threat modeling)
  - Nie ma sensu – prościej jest użyć gotowego modelu dla aplikacji webowej (pod warunkiem, że taki akurat istnieje)
  - Selekcja oczywistych obszarów kodu do audytu

AVET
Information and Network Security

Przed audytem kodu…

1. Ustalenie procedur dostępu do kodu źródłowego
2. Ustalenie procedur zgłaszania i naprawiania błędów
   1. **Każdy błąd wymaga naprawy**
   2. Nie ma znaczenia czy podatność jest w tym momencie do wykorzystania czy też nie – każda jest naprawiana
   3. **Podatności których nie można naprawić w prosty sposób na poziomie kodu źródłowego zostaną zaadresowane tymczasowo przez inne zabezpieczenia**
3. Uporządkowanie repozytorium kodu
4. Checkout z repozytorium do audytu
5. Zastąpienie tradycyjnych ticketów zgłoszeniami z systemu Barracuda
6. **Programiści nie używają żadnego narzędzia do statycznej lub dynamicznej analizy kodu**
7. **Przystosowanie platformy SecureCode do wymagań projektu**

AVET
Information and Network Security

Gotowy model zagrożeń

- Walidacja danych wejściowych / wyjściowych
- Autoryzacja i uwierzytelnienie użytkowników
- Zarządzanie sesją i stanem
- Styki różnych komponentów
  - Serwer aplikacyjny – baza danych
  - Serwer HTTP – serwer aplikacyjny
- Konfiguracje różnych komponentów
- Standard OWASP Top 10 – dobry punkt wyjścia, ale nie złoty środek

Walidacja danych wejściowych

Podstawowe zasady

- Nigdy nie ufaj danym z zewnętrznego (niezaufanego) źródła
- Nigdy nie przetwarzaj stanu / sesji po stronie użytkownika
- Nigdy nie ufaj ograniczeniom formularzy

Obszary

- Formularze
- Protokół aplikacyjny
  - Zapytania GET, POST
  - Pozostałe metody
  - Cookies
  - Trzymanie stanu
  - Sesja (ze szczególnym uwzględnieniem jej kończenia)

Walidacja danych - problemy

- Procedury walidacji danych nie są łatwe i szybkie do napisana (zwłaszcza danych wyjściowych)
- Część stanu zawsze jest przetwarzana po stronie użytkownika:
  - Ajax
  - ActiveScript
  - JavaScript

AVET
Information and Network Security

Styk różnych komponentów: przypadek MySQL

Typowe problemy

- Konto aplikacyjne ze zbyt wysokimi uprawnieniami
- Brak adekwatnej ochrony interfejsów sieciowych
- Brak fuzzingu na poziomie zapytań SQL

Przykładowe zapytania

- CVE-2008-3963:
  - select b'';
- CVE-2010-2008:
  - alter database `#mysql50#.`
  - alter database `#mysql50#../`
- CVE-2009-2446:
  - %s%s%s%s%s
- CVE-2010-3833:
  - LEAST()
  - GREATEST()

Styki różnych komponentów: przypadek MS SQL Server

Typowe obszary zagrożeń

- Użytkownik aplikacyjny ze zbyt wysokimi uprawnieniami
- Brak adekwatnej kontroli intefejsów sieciowych
- Procedury składowe
- Rozszerzone procedury składowe
- Nadal nadużywanie konta „sa”
- Nadal zakładanie że konto „sa” jest bez hasła

Przykłady „nietypowych” ataków

- SQL Server 2000: 3 bajtowy patch wyłączający kontrolę dostępu
- Używanie procedur składowych do dalszego ataku
- Przepełnienia bufora w procedurach składowych

Konfiguracje różnych komponentów: przypadek PHP

- allow_url_fopen
- disable_functions
- display_errors
- enable_dl
- error_reporting
- file_uploads
- log_errors
- magic_quotes_gpc
- memory_limit
- open_basedir
- register_globals
- Safe_mode

Przypadek #2: Stworzenie modelu bezpieczeństwa

ZAPROJEKTUJ OD POCZĄTKU JAKO BEZPIECZNY SYSTEM

AVET
Information and Network Security

Podstawowe założenia

- Biznes rozumie rolę bezpieczeństwa
- Compliance jest istotnym elementem dla projektowanego systemu
- Projekt ma zdefiniować pojęcie „bezpiecznego systemu” już na etapie projektu
- Gotowa, wysokopoziomowa lista kontrolna dla audytu

Podstawowe ryzyka projektowe

- Dostawca aplikacji posiada już pewną bazę kodu i architekturę, która nie była projektowana zgodnie z wynikiem projektu
- Koszty i czas wdrożenia modelu – umowa została podpisana z Dostawcą przed podpisaniem modelu
- Edukacja biznesu – biznes musi zrozumieć pewne istotne elementy zarządzania bezpieczeństwem
- Edukacja ekspertów ds. bezpieczeństwa - muszą oni zrozumieć założenia biznesu

AVET
Information and Network Security

Model zagrożeń

- Założenia biznesowe zmapowane na ryzyko biznesowe
- Ryzyko biznesowe definiuje zagrożenia biznesowe
- Zagrożenia biznesowe mają wpływ na zagrożenia techniczne
- Różne modele zagrożeń ze względu na wykorzystanie cienkiego i grubego klienta
- Więcej niż jeden profil użytkownika

Przypadek #3: System developerski

DAJ SZANSĘ DEVELOPEROM

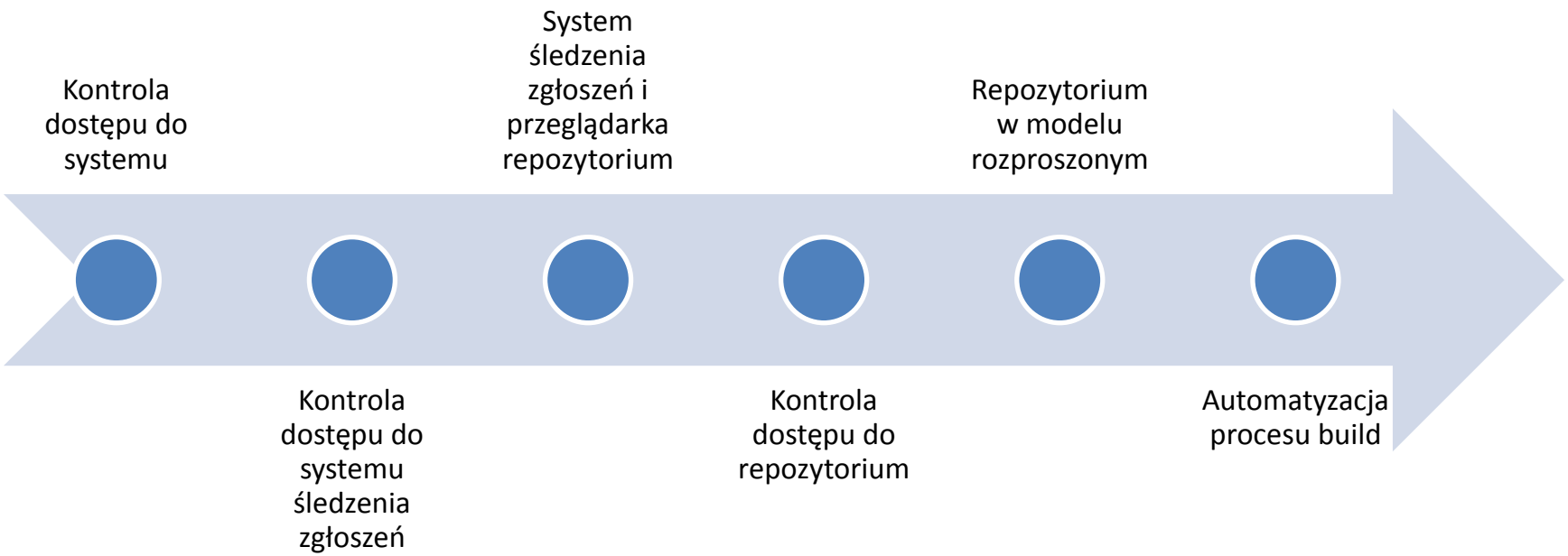
AVET
Information and Network Security

Podstawowe założenia

- Biznes rozumie rolę bezpieczeństwa
- Produkowane oprogramowanie ma być bezpieczne
- Developerzy mają mieć zapewnione środowisko, które spełni oczekiwania biznesu
- Wykorzystanie metodyki Agile
- XP odpadło z kilku powodów: najważniejszym był brak własności wybranych obszarów kodu źródłowego

AVET
Information and Network Security

Ogólna architektura z wykorzystaniem koncepcji stepping stone



Rozproszony system kontroli wersji

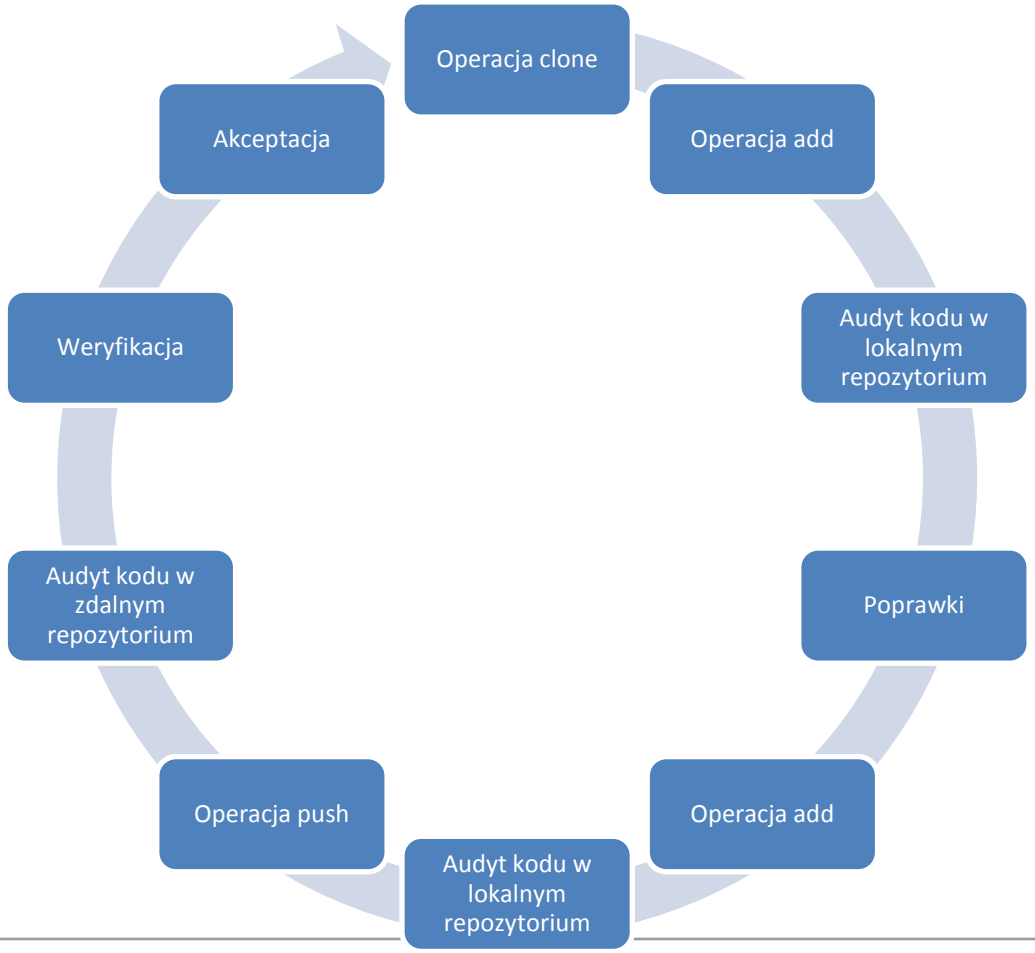
Zalety

- Łatwość zapewnienia ciągłości
- Łatwość pracy w chmurze
- Łatwość współpracy z systemem

Wady

- Odmiejscowienie repozytorium
- Wymagana dodatkowa dyscyplina

Statyczny audyt kodu



AVET
Information and Network Security

Inne istotne aspekty

- Zestaw reguł tworzenia kodu
- Unit-testy
- Scenariusze testowania
- Proces testów

Zmiana paradygmatu bezpieczeństwa

BEZPIECZEŃSTWO APLIKACYJNE W CHMURZE

AVET
Information and Network Security

Ważne pytania

- Czy rodzaj chmury ma znaczenie dla bezpieczeństwa?
- Czy technologia chmury ma znaczenie dla bezpieczeństwa?
- Czy technologia aplikacji ma znaczenie dla bezpieczeństwa?
- Czy aplikacje tradycyjne „wyrzucone” do chmury stają się bardziej (nie)bezpieczne?
- Czy aplikacje tworzone z myślą o chmurze są bezpieczniejsze od tradycyjnych?
- Compliance…

AVET
Information and Network Security

Zmiana paradygmatu bezpieczeństwa

- Jak wygrać wojnę jeśli nie mamy nawet komputera?
  - Kto będzie miał komputer: Amazon, IBM, NASA, NSA…
- Przetwarzanie rozproszone: penetracja jednego miejsca oznacza penetrację wszystkich punktów?
- Patch management
- Wirtualizacja

AVET
Information and Network Security

Podsumowanie

- Wszystkie komponenty aplikacji oraz środowiska wykonania są równie ważne
- Procedury wytwarzania i zarządzania konfiguracją oprogramowania są równie ważne co procedury eksploatacyjne
- Do incydentu i podatności trzeba się przygotować wcześniej

#20	Setup email address for security team	postfix	closed	done

AVET
Information and Network Security

Dziękuję za uwagę

- Pytania?

aleksander.czarnowski@avet.com.pl

AVET
Information and Network Security