Jak efektywnie wykrywać podatności bezpieczeństwa w aplikacjach?

dr inż. Jakub Botwicz
CISSP, ECSA, GWAPT

jakub.botwicz@gmail.com

OWASP
19.11.2014

The OWASP Foundation
http://www.owasp.org

Wykrywanie podatności bezpieczeństwa

- Testy bezpieczeństwa (penetracyjne)
  - White box vs. Grey box vs. Black box
  - Aplikacji i/lub infrastruktury
- Analiza dynamiczna aplikacji
  - Testy interfejsów webowych lub webserwisowych
  - Monitorowanie działania aplikacji
- Analiza statyczna aplikacji
  - Manualne przeglądy kodu lub binariów
  - Analiza automatyczna kodu lub plików binarnych
- Programy *bug bounty*

ANALIZA DYNAMICZNA

Skanery podatności aplikacji webowych

Fazy działania:

I. Rozpoznanie aplikacji
(znalezienie dostępnych adresów URL i parametrów)

II. Próby ataków
(użycie np. fuzzingu)

III. Weryfikacja wyników
(sprawdzenie czy podatności rzeczywiście istnieją)

IV. Zebranie wyników

Ograniczenia skanerów automatycznych

- Niestandardowe sposoby komunikacji aplikacji
  - Skanery nie „rozumieją” formatu wiadomości
- Operacje inicjowane w sposób niestandardowy
  - Skanery nie potrafią wywołać operacji
- Walidacja danych
  - Skanery nie „rozumieją” komunikatów o błędach
- Formularze wielostopniowe
  - Skanery nie potrafią dokończyć operacji
- Zabezpieczenia przeciw automatom
  - CAPTCHA, uwierzytelnienie SMS lub tokenem

Testy manualne vs. Testy automatyczne

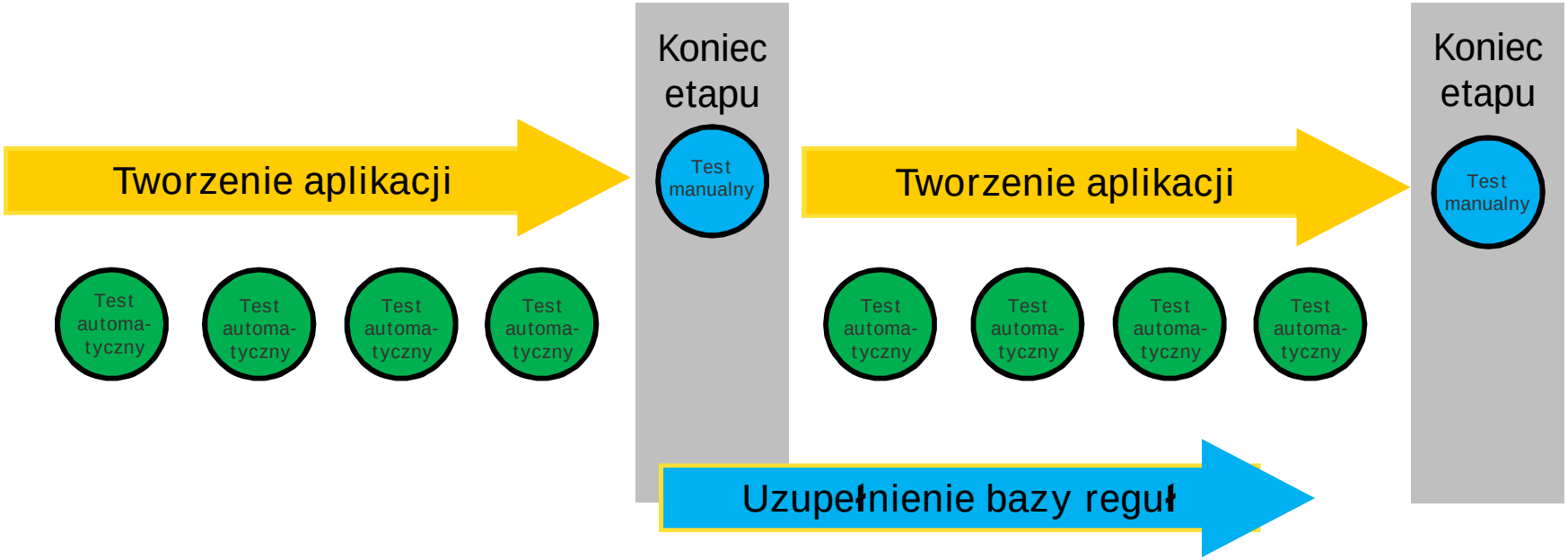
Czynniki wpływające na jakość wyników:

- Doświadczenie testera
- Czas dostępny na testy
- Zmęczenie testera aplikacją
- Jakość narzędzia
- Odpowiednia konfiguracja narzędzia
- Dopasowanie narzędzia do aplikacji

Zalety:

- Lepsze zrozumienie aplikacji i logiki biznesowej
- Wykrywanie nietypowych podatności
- Grupowanie podatności w scenariusze ataku
- Niższe koszty
- Krótszy czas testowania
- Testowanie poza godzinami pracy
- Powtarzalność wyników
- Możliwość częstego powtarzania

Połączenie testów automatycznych i manualnych



- Testy automatyczne wykonywać często i regularnie (regresja)
- Testy manualne
  - wykonywać przy istotnych zmianach
  - na podstawie ich wyników uaktualniać testy automatyczne

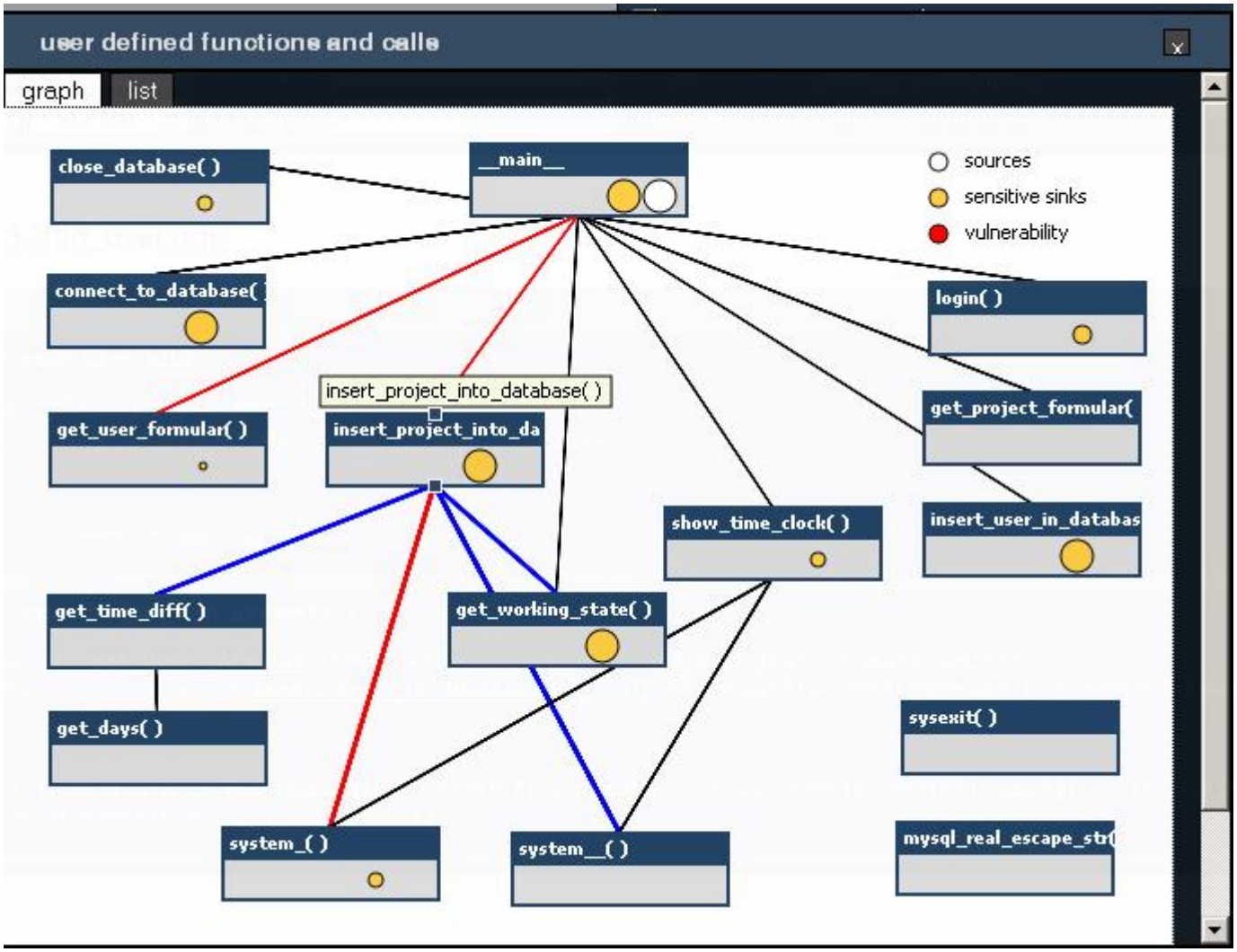
ANALIZA STATYCZNA

Techniki działania narzędzi automatycznych

- Wyszukiwanie wzorców
  - podejrzane funkcje – Random, gets(), MD5, DES
  - słowa kluczowe – password
- Analiza „source to sink”
  - wejście – interfejs webowy
  - wyjście – baza danych
  - podatność – SQL Injection

    inne podatności: XSS, większość Injection
- Metryki
  - Złożoności funkcji lub gęstości komentarzy

Analiza „source to sink” – RIPS



Analiza statyczna vs. Analiza dynamiczna

(Przegląd kodu lub binariów) (Testy penetracyjne)

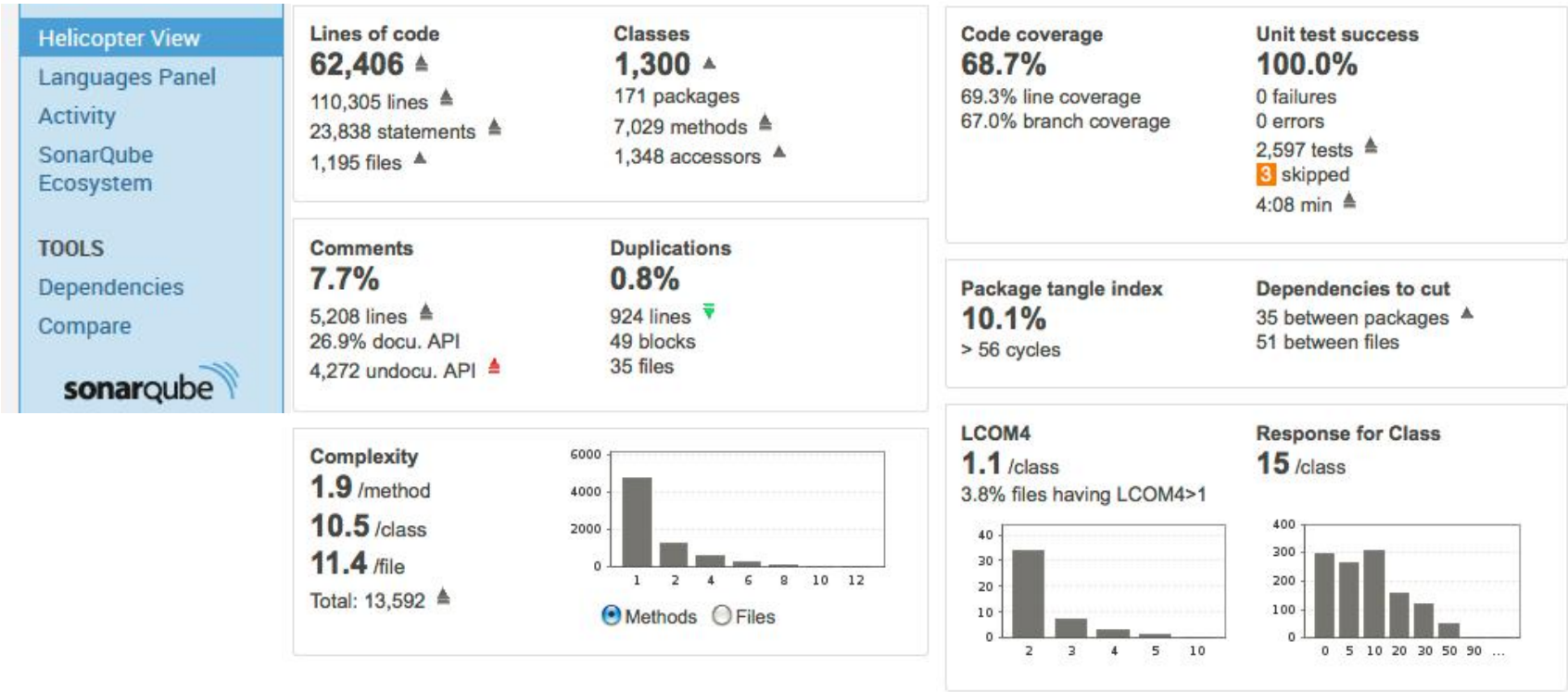
Łatwość analizy:

- Sposobu przechowywania danych w bazie
- Szyfrowania danych
- Komunikacji z wewnętrznymi systemami
- Logowania zdarzeń
- Obsługi interfejsów wejściowych
- Logiki biznesowej aplikacji

Problemy:

- Trudniej jest zweryfikować czy określony kod jest używany (więcej błędów false-positive)
- Trudniej jest zweryfikować istniejące ale nieskuteczne mechanizmy

Continous Inspection – SonarQube



Programy *bug bounty*

- Trudności z oszacowaniem kosztów programu
  - Płacimy za rzeczywiste znalezione podatności
  - Musimy obsłużyć wszystkie otrzymane zgłoszenia
- W produkcyjnym systemie pojawiają się testerzy
  - Jak odróżnić *bug bountera* od włamywacza?
  - Poszukiwacze błędów generują dodatkowy ruch
- Co będzie po znalezieniu poważnego błędu?
  - Czy nie będzie prób szantażu?

Monitorowanie działania aplikacji

- Analiza
  - zdarzeń (logów) – SIEM
  - ruchu sieciowego – IDS/IPS
  - ruchu webowego – WAF
- Techniki
  - Wykrywanie anomalii
  - Analiza powłamaniowa
  - Debuggowanie lub profilowanie aplikacji

Testy na poziomie infrastruktury

- Konfiguracja serwerów SSL/TLS
- Niezabezpieczone konsole administracyjne
- Nieużywane moduły aplikacji
- Niezabezpieczone usługi systemów operacyjnych
  - FTP
  - SMTP

Dziękuję za uwagę!

Czy mają Państwo jakieś pytania?