Fuzzing

Piotr Łaskawiec
J2EE Developer/Pentester

Metrosoft (www.metrosoft.com)
piotr.laskawiec@gmail.com

OWASP
14.01.2010

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

Abstract

- Metody testowania aplikacji
- Zapewnienie bezpieczeństwa aplikacji
- Fuzzing – definicja
- Zastosowanie fuzzingu
- Podział fuzzerów
- Fuzzing a SDLC
- Kto korzysta z fuzzerów?
- Przykładowe fuzzery
- Web application fuzzing
- Podsumowanie

Testowanie aplikacji

- Popularne testy:
  - Testy jednostkowe
  - Testy funkcjonalne
  - Testy regresyjne
  - Testy wydajnościowe
  - Testy usability
- Inna klasyfikacja:
  - Whitebox, Graybox, Blackbox
- Co z bezpieczeństwem tworzonej aplikacji?
  - Bezpieczeństwo na poziomie projektowania, implementacji, testowania oraz wdrażania.

Zapewnianie bezpieczeństwa aplikacji

- Reagowanie vs zapobieganie
- Reagowanie:
  - Firewall
  - IDS/IPS
  - Antywirus
  - Mechanizmy uwierzytelniające
  - Skanery podatności (Nessus, Nikto, etc.)
  - Itd.
- Zapobieganie:
  - Fuzzing!
  - Audyt kodu/RE

Fuzzing - definicja

- Fuzzing jest metodą testowania oprogramowania pod kątem występowania luk w bezpieczeństwie oraz nieprzewidzianych reakcji programu, za pomocą częściowo losowych (pseudolosowych) danych.
- Fuzzing najczęściej jest procesem w pełni zautomatyzowanym - „uruchom i czekaj na wyniki”.

Fuzzing – co to znaczy w praktyce?

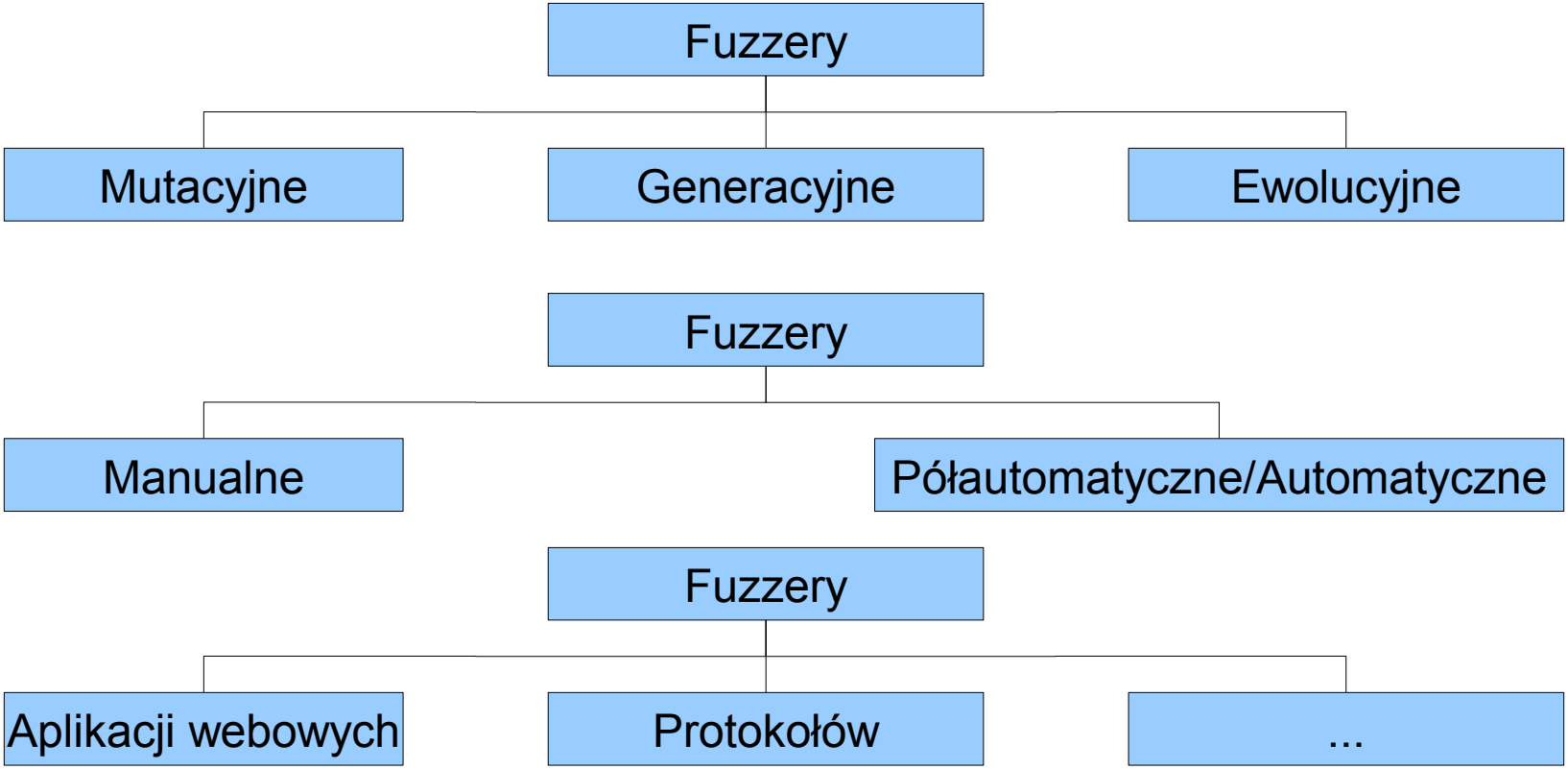
- Fuzzing == Negative testing
- Celem fuzzera jest przekazanie do testowanej aplikacji wadliwych (zbyt długi ciąg znaków, niepoprawne kodowanie, wadliwy format pliku, zła kolejność komunikatów) danych.
- Liczymy na zaakceptowanie wadliwych danych i wystąpienie nieprzewidzianej reakcji programu – DoS, wyświetlenie komunikatu o błędzie, zwiększenie zapotrzebowania na zasoby.
- Naszym celem jest „zepsucie” aplikacji!

Zastosowanie fuzzerów

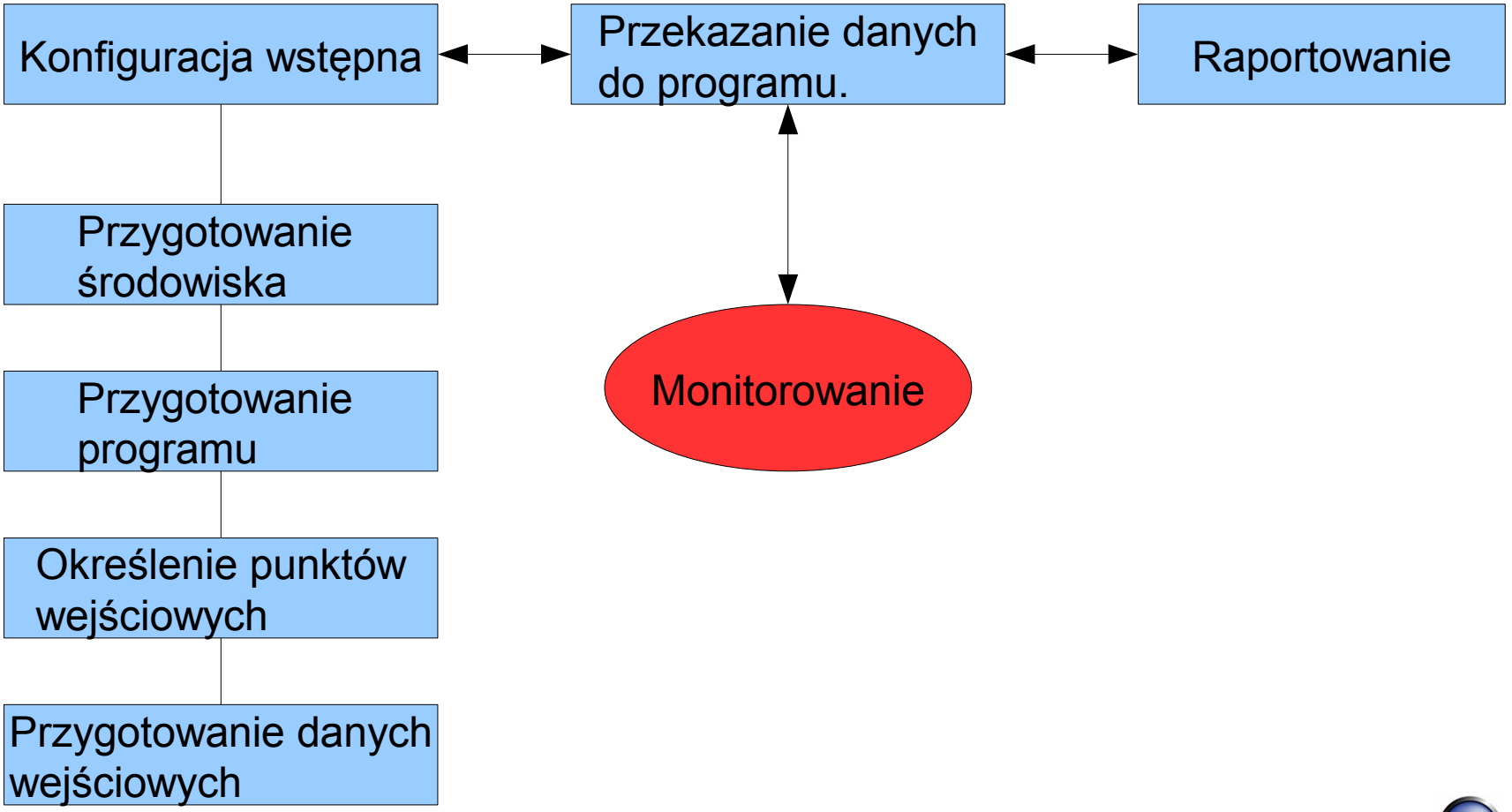
- Aplikacje działające lokalnie
- Aplikacje webowe
- WebService'y
- Aplikacje sieciowe
- Kontrolki ActiveX
- Pliki
- Biblioteki
- ...

Klasyfikacja fuzzerów

- Istnieje wiele kryteriów klasyfikacji fuzzerów
- Przykłady:



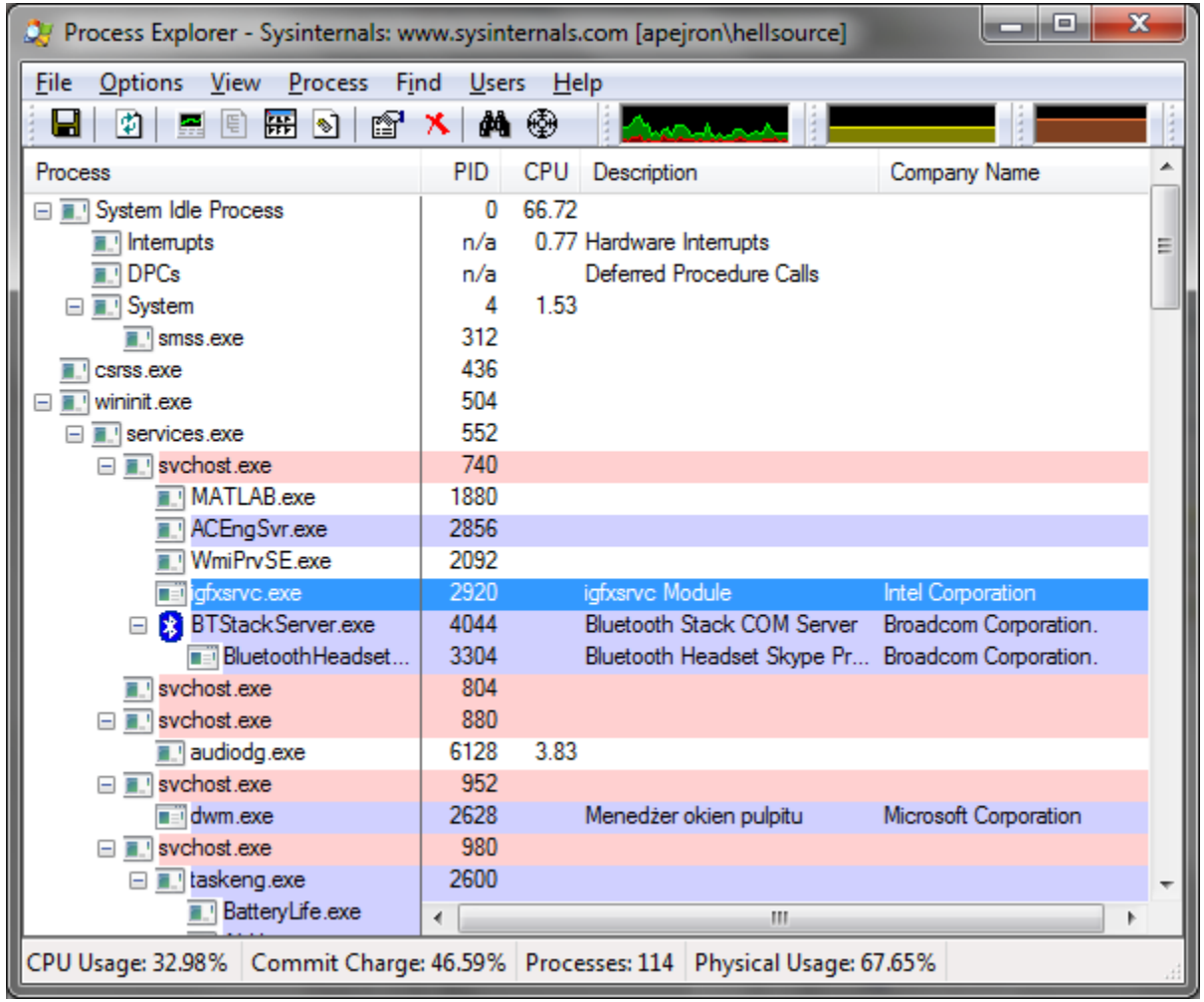
Proces fuzzingu



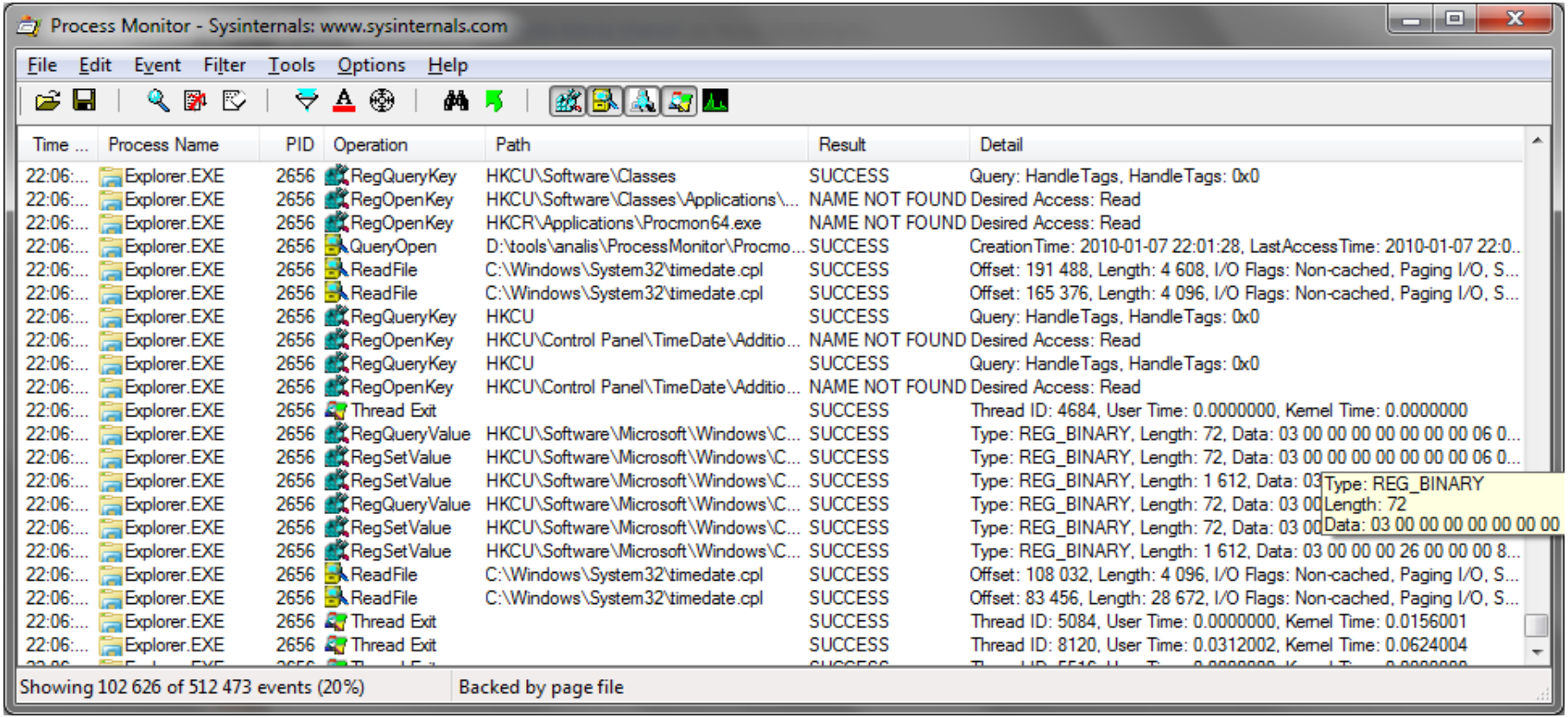
Monitorowanie testowanej aplikacji

- Obserwacja zachowania programu
- Logi systemowe
- Debuggery (!exploitable...)
- Monitory procesów, plików, połączeń sieciowych
- Wirtualizacja (VMWare)
- Modyfikacja kodu źródłowego (dodanie punktów kontrolnych)
- Inne techniki (Valgrind, Guard Malloc)
- Techniki łączone

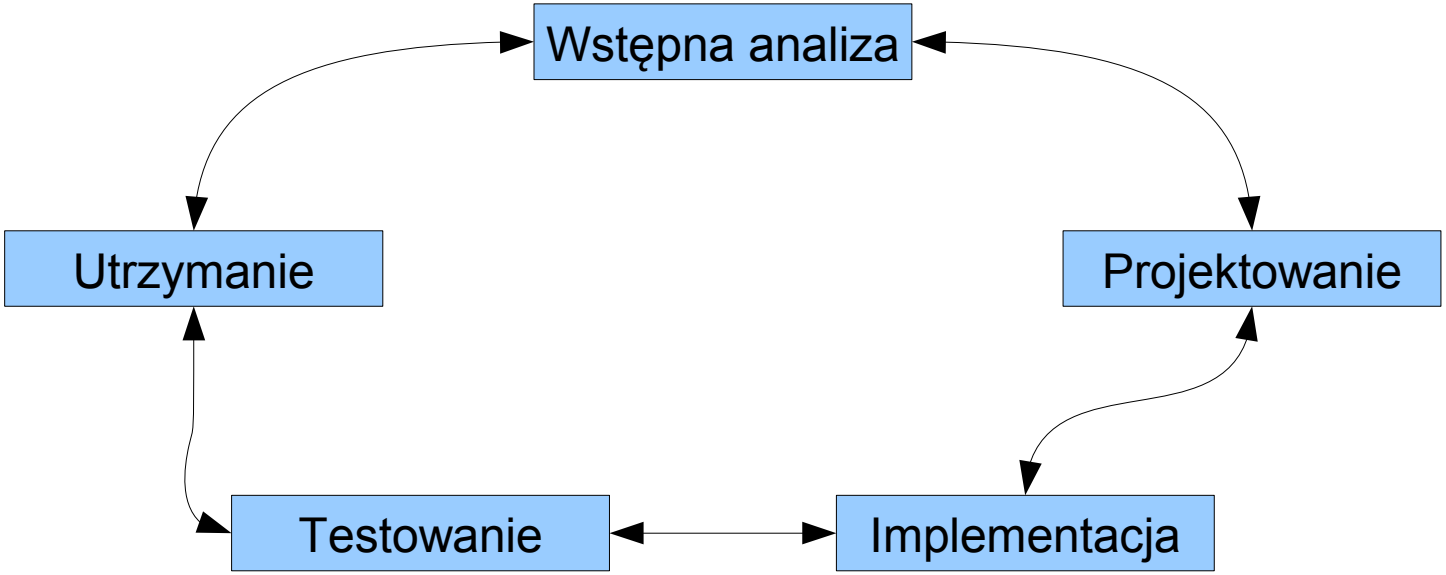
Process Explorer



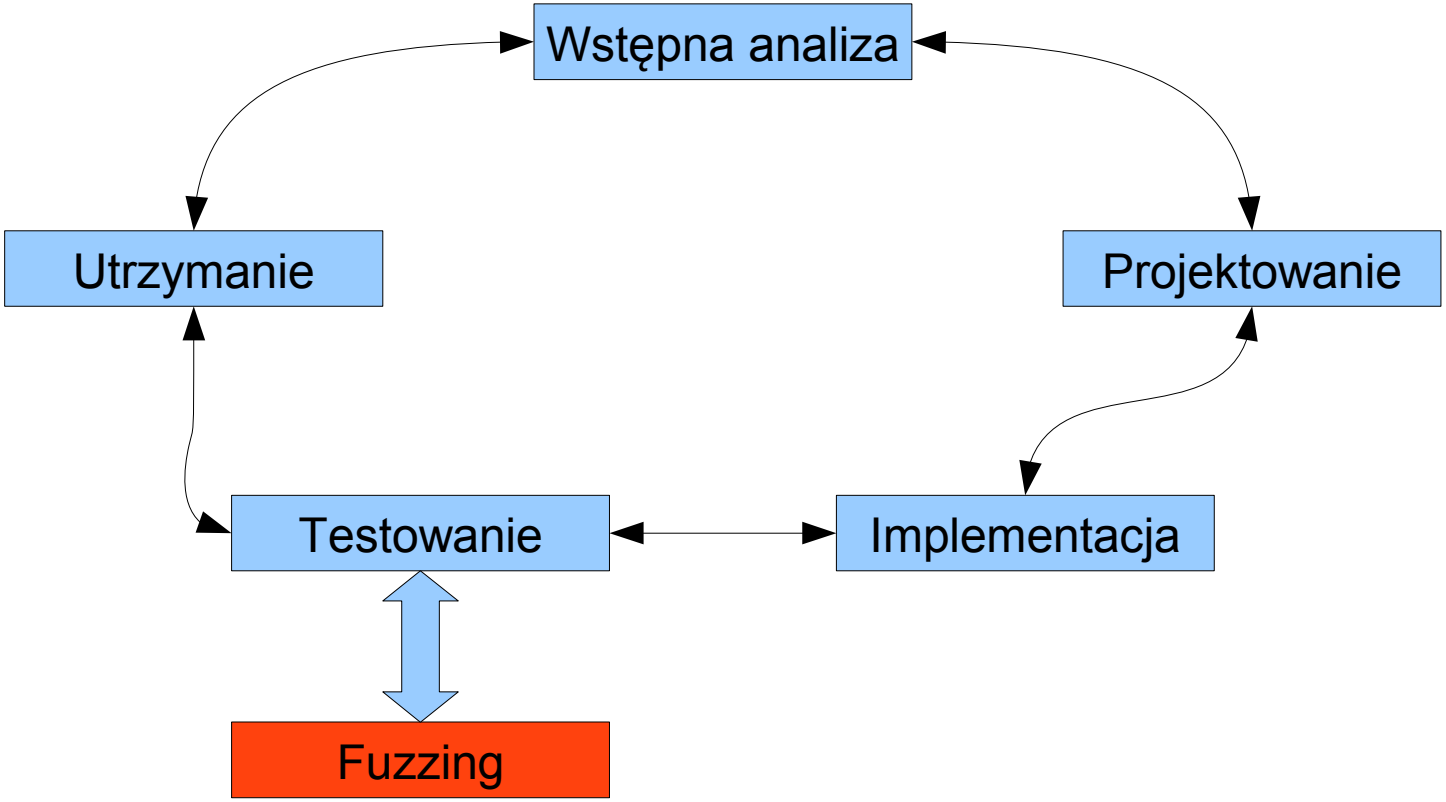
Process Monitor



Fuzzing a SDLC



Fuzzing a SDLC



Fuzzing a SDLC

- Po opublikowaniu nowej wersji, aplikacja jest testowana przez przygotowane wcześniej fuzzery.
- Wyniki testów weryfikowane są przez testerów i trafiają w ręce programistów.
- W razie wystąpienia błędów programiści poprawiają oprogramowanie.
- Nowy build jeszcze raz przechodzi przez proces fuzzingu.

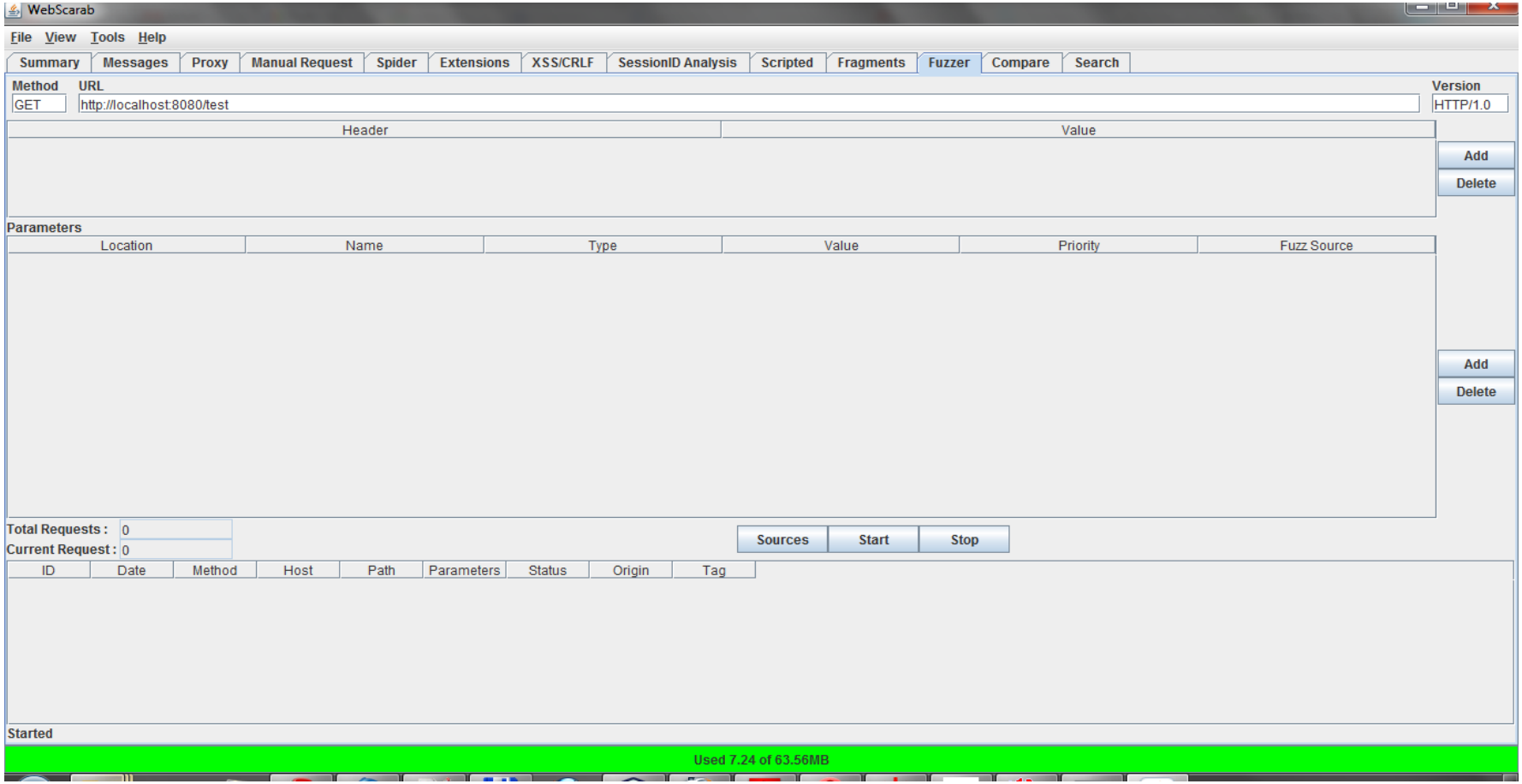
Kilka przykładów

- Adobe
  - http://blogs.adobe.com/asset/2009/12/fuzzing_reader_-_lessons_learned.html
- Błędy w IIS
  - http://pentestit.com/2009/12/28/microsoft-iis-day-open/
- Mozilla JavaScript fuzzer
  - http://blog.mozilla.com/security/2007/08/02/javascript-fuzzer-available/
- Microsoft MiniFuzz
  - http://www.microsoft.com/downloads/details.aspx?FamilyID=b2307ca4-638f-4641-9946-dc0a5abe8513&displaylang=en
- Google Flayer
  - http://code.google.com/p/flayer/

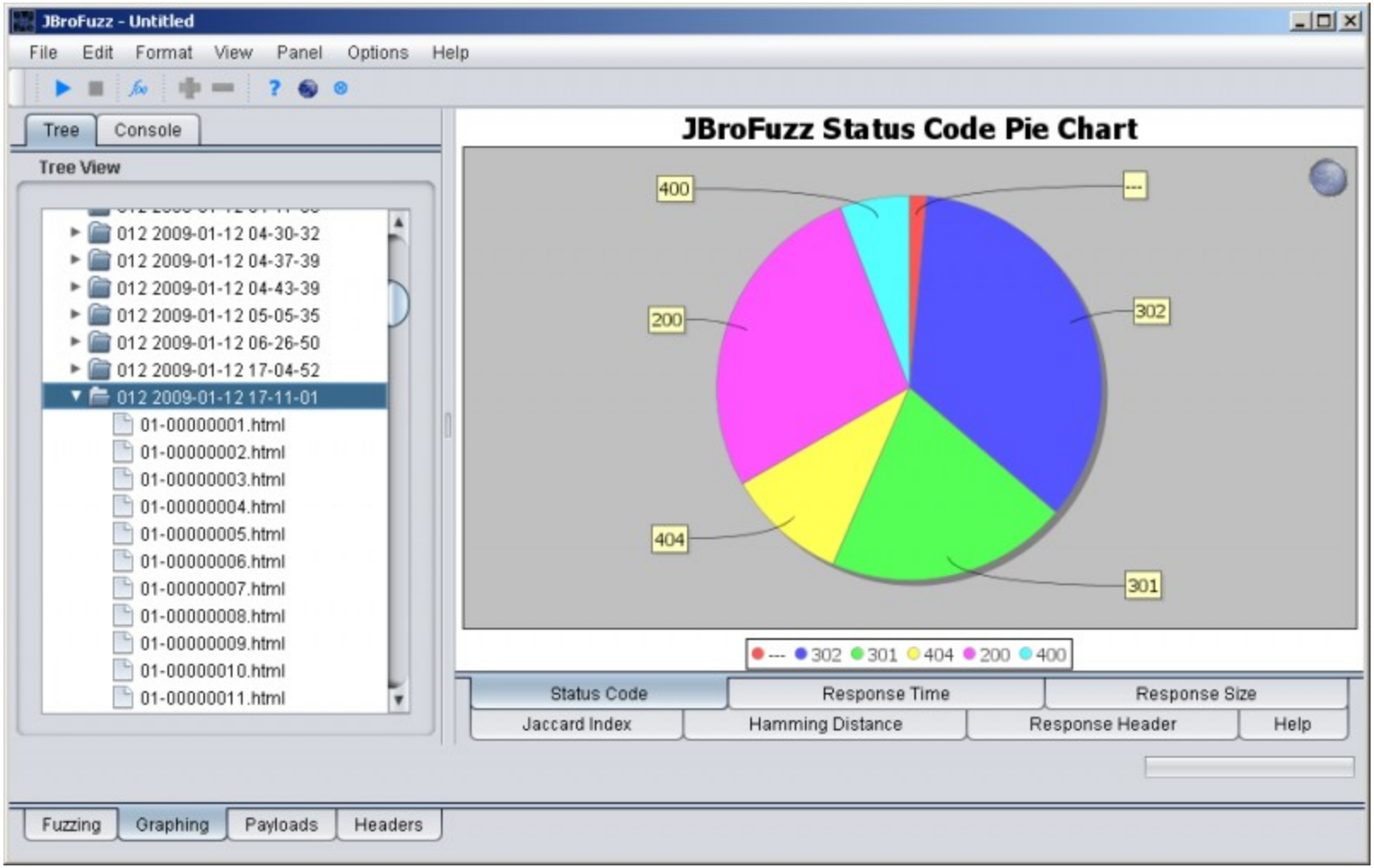
Przykładowe fuzzery

- Frameworki:
  - Peach (http://peachfuzzer.com/)
  - Sulley
- Wyspecjalizowane fuzzery:
  - JBroFuzz (OWASP)
  - WSFuzzer (OWASP)
  - TAOF
  - Wfuzz
  - Spike Proxy
  - WebFuzz
- Autorskie rozwiązania

WebScarab Fuzzer plugin



JBroFuzz


JBroFuzz - Untitled
File Edit Format View Panel Options Help
Tree Console
Tree View
012 2009-01-12 04-30-32
012 2009-01-12 04-37-39
012 2009-01-12 04-43-39
012 2009-01-12 05-05-35
012 2009-01-12 06-26-50
012 2009-01-12 17-04-52
012 2009-01-12 17-11-01
01-00000001.html
01-00000002.html
01-00000003.html
01-00000004.html
01-00000005.html
01-00000006.html
01-00000007.html
01-00000008.html
01-00000009.html
01-00000010.html
01-00000011.html
JBroFuzz Status Code Pie Chart
400

200
302
404
301
--- 302 301 404 200 400
Status Code
Response Time
Response Size
Jaccard Index
Hamming Distance
Response Header
Help
Fuzzing
Graphing
Payloads
Headers

Fuzzing aplikacji webowych

- Problemy:
  - Identyfikowanie punktów wejściowych
    - Analiza komunikacji HTTP
    - Webspidering
    - Wyszukiwarki internetowe
  - Generowanie danych testowych
    - Payloady zapisane w programie
    - Bruteforce
    - Generowanie danych na podstawie wzorców
  - Identyfikowanie błędów

Identyfikowanie błędów

- Kody odpowiedzi HTTP
- Analiza treści strony
- Porównywanie struktury strony
- Ataki czasowe
- Wielokrotne zapytania
- Analiza danych jednoznacznie identyfikujących daną stronę
- Logi

Anty-fuzzing

- Nie można bronić się bezpośrednio przed fuzzingiem!
- Obrona ogólna:
  - Walidacja danych wejściowych
  - Stosowanie się do dobrych praktyk programistycznych
  - Dbanie o bezpieczeństwo oprogramowania przez wszystkie fazy SDLC

Podsumowanie

Zalety fuzzingu

- Pełna automatyzacja (w większości wypadków)
- Fuzzery znajdują realne podatności
- Możliwość wykrycia błędów trudnych do znalezienia poprzez manualne testy
- Możliwość szybkiego uzyskania zadowalających wyników (pierwszego błędu)

Wady fuzzingu

- Brak możliwości wykrycia błędu logicznego
- Brak możliwości wykrycia bardzo złożonych podatności (gdzie efekt końcowy jest składową sekwencji operacji)
- Trudny do sprecyzowania czas potrzebny na przeprowadzenie testów

Informacje dodatkowe

- Prezentacje:
  - PyCON 2008
  - SEConference 2009
- Strony:
  - fuzzing.eu
  - fuzzing.org
  - krakowlabs.com/lof.html

09-10.04.2010

www.seconference.pl

Pytania

Dziękuję za uwagę!