

Zautomatyzowane testowanie aplikacji webowych

Agenda

- Automate tests of applications - introduction
- Overview of available solutions
- Weaknesses of traditional scanners
- Scanners based on mimicking user behavior
- How to build your own scanner?
- Methods of error detection
- Future – automatic pentester?
- Summary

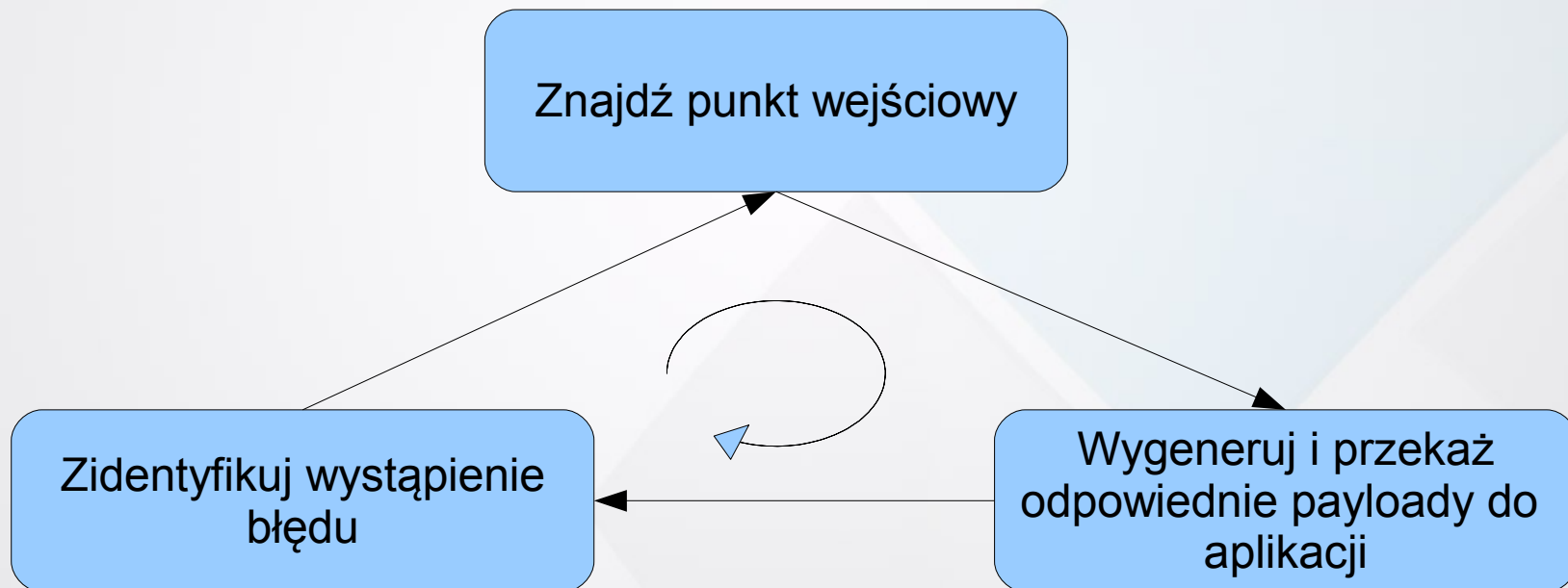
Błędy w aplikacjach webowych

- Cross Site Scripting
- Cross Site Request Forgery
- Injection Bugs (SQL Injection, LDAP Injection...)
- Bad Session Management
- Remote File Execution
- Logical Bugs
- ...

Automatyzacja

- Automatyzacja powtarzalnych procesów jest zupełnie naturalna
- Każdy proces składający się z sekwencji wielu powiązanych, niezmiennych czynności warto zautomatyzować! (oszczędność czasu, pieniędzy)
- Czy testy bezpieczeństwa aplikacji webowej można wykonywać w sposób automatyczny?

Testowanie aplikacji webowej - schemat



Popularne narzędzia

- WebFuzz
- Nikto
- Powerfuzzer
- Wfuzz
- Nessus
- Skipfish
- sqlmap
- ...

Złożoność aplikacji webowych

- Aplikacje webowe stają się coraz bardziej rozbudowane.
- Logika aplikacji nie opiera się jedynie na pojedynczych parach „odpowieź-żądanie”.
- Oprogramowanie posiada stan.
- Uzyskanie oczekiwanego wyniku wymaga wykonania sekwencji operacji.

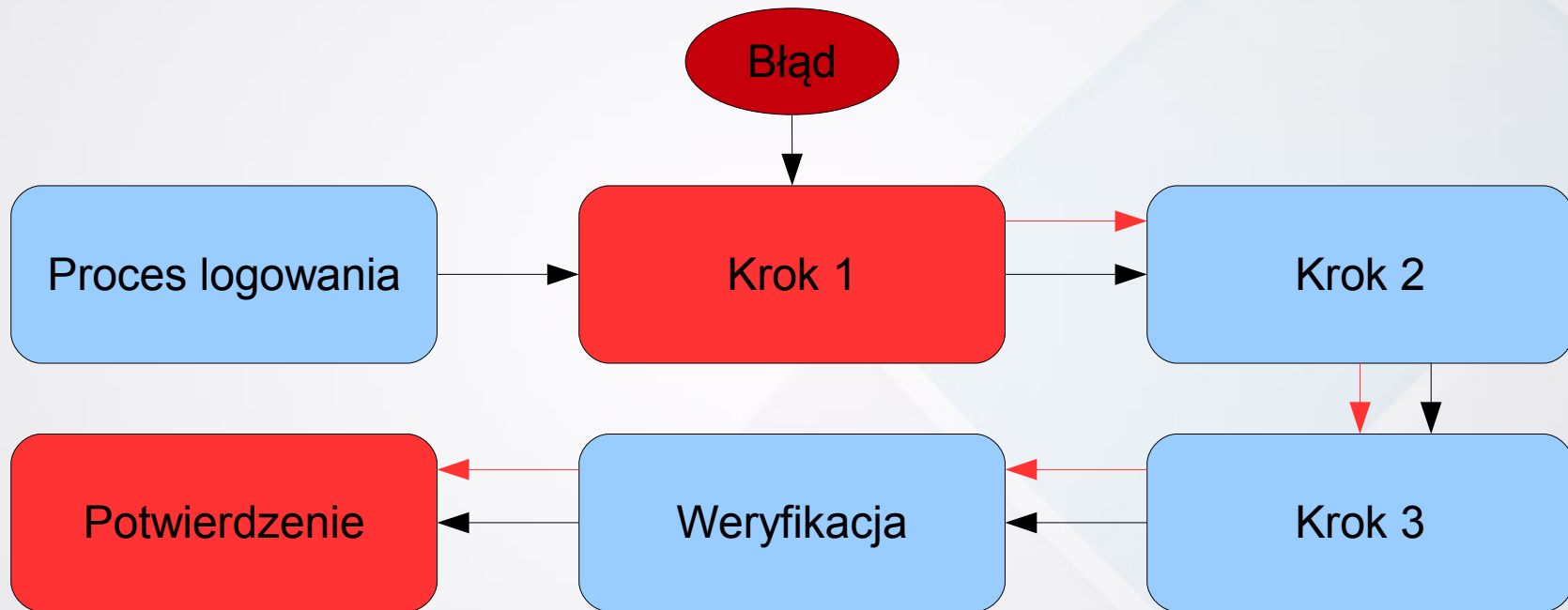
Złożoność aplikacji webowych



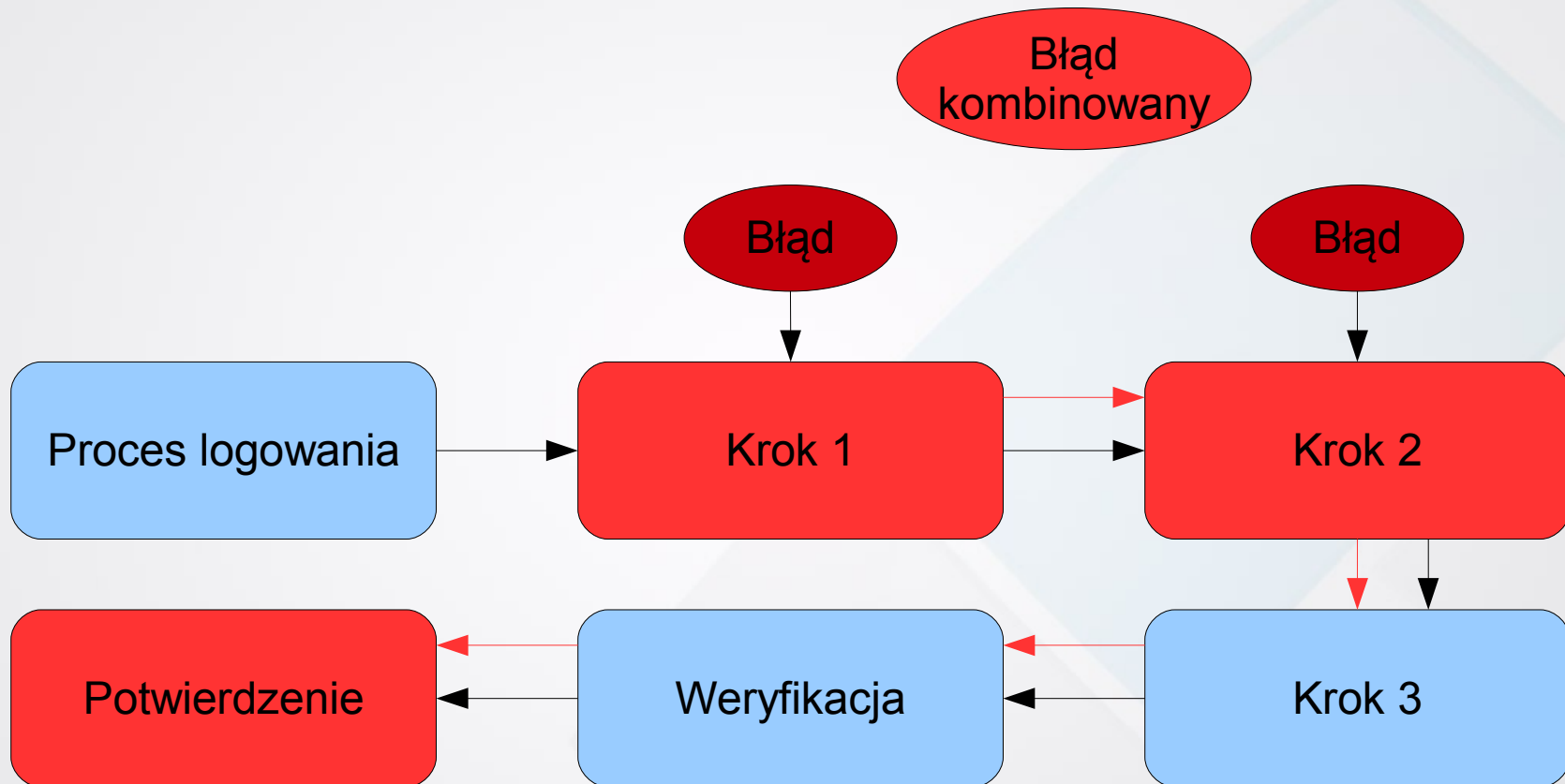
Przykłady:

- Wirtualny koszyk
- Składanie transakcji
- Testy
- ...

Złożoność aplikacji webowych

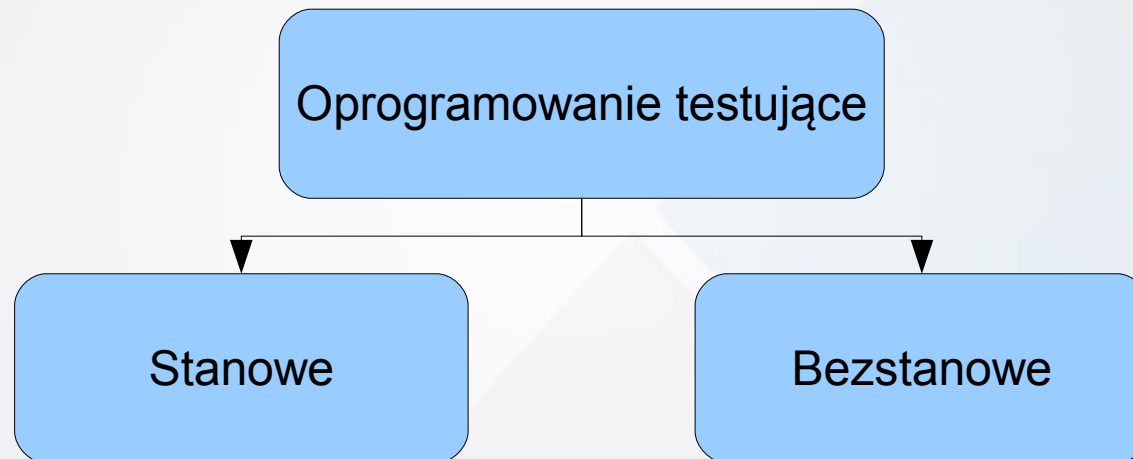


Złożoność aplikacji webowych



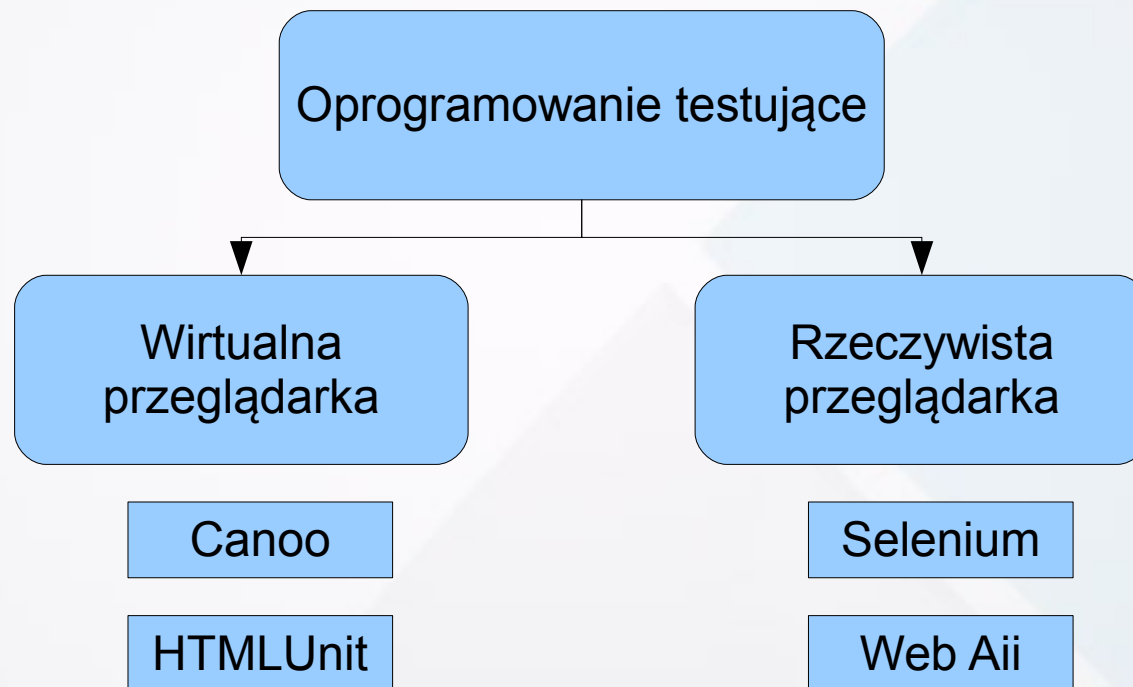
Przykład

Narzędzia testujące



Potrzeba wprowadzenia oprogramowania testującego opartego na scenariuszach.

Narzędzia testujące



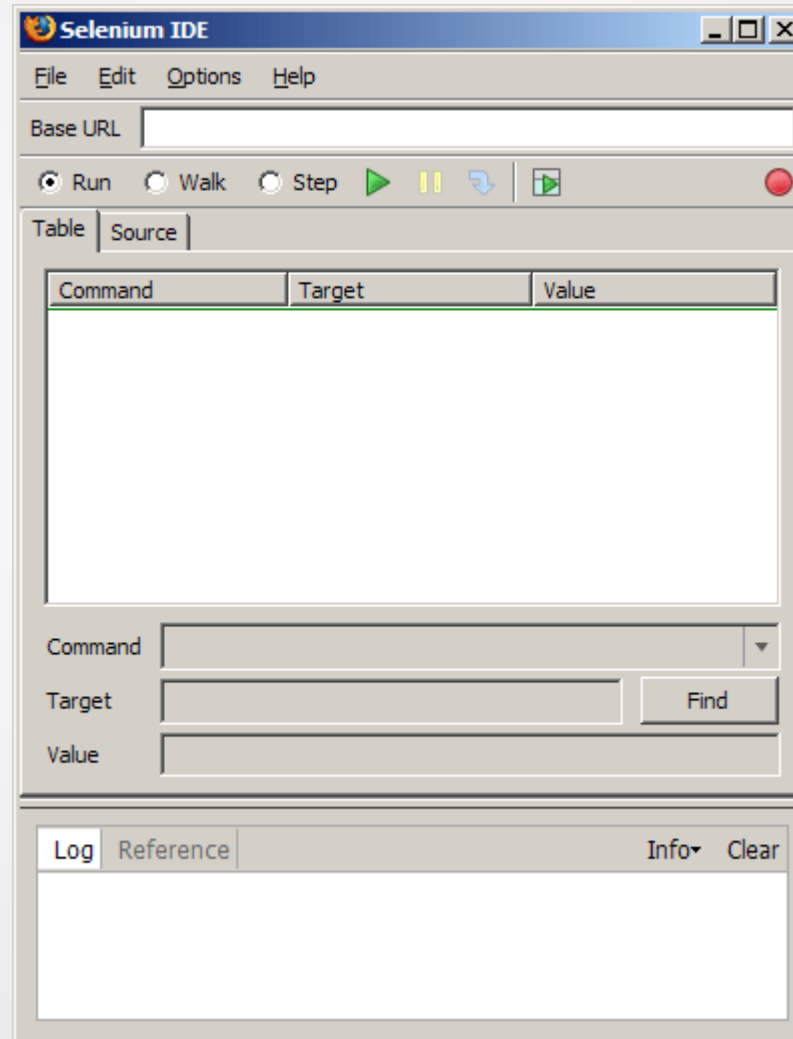
Selenium

- Oprogramowanie automatyzujące testy aplikacji webowych
- Składa się z wielu komponentów: Selenium IDE, Selenium Remote Control, Selenium Grid
- Integracja z wieloma językami programowania
- Ma zastosowanie przy testowaniu aplikacji pod kątem występowania luk w bezpieczeństwie
- Open-source

Selenium

```
public void runTest() throws Exception
{
    setUp("http://www.example.com/", "*firefox");
    List<String> xssPayloads = getPayloads();
    for(String s : xssPayloads)
    {
        selenium.open("/");
        selenium.type("test", s);
        selenium.click("confirmButton");
        selenium.waitForPageToLoad("10000");
        checkForXss(selenium.getHtmlSource());
    }
}
```

Selenium IDE



Web Aii

- Darmowa biblioteka .NET
- Świetna integracja z IDE Visual Studio
- Możliwość testowania aplikacji Silverlight
- Bardzo dobre wsparcie dla drag&drop oraz AJAX
- Brak otwartych źródeł

Web Aii

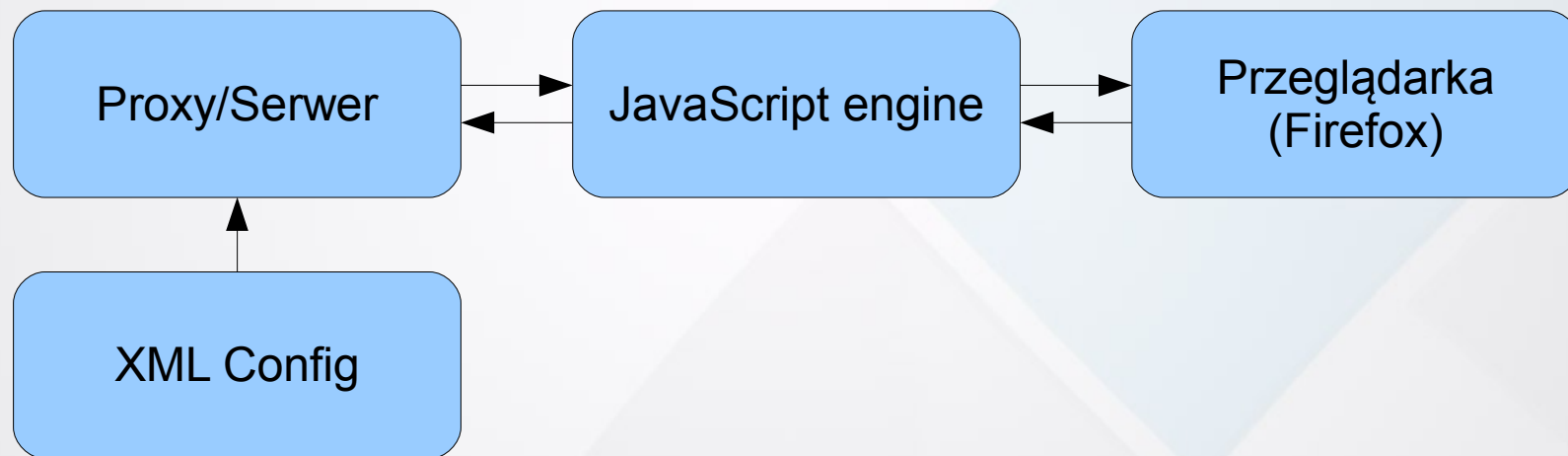
```
void runTest()
{
    List<string> xssPayloads = getPayloads();
    Settings mySettings = new Settings(BrowserType.InternetExplorer, @"D:\webtests\log\");
    Manager myManager = new Manager(mySettings);
    myManager.Start();
    myManager.LaunchNewBrowser();

    foreach(String s in xssPayloads)
    {
        ActiveBrowser.NavigateTo("http://www.example.com");
        Element input1 = ActiveBrowser.Find.ById("test");
        ActiveBrowser.Actions.SetText(input1, s);
        HtmlInputSubmit submit = form.Find.ById<HtmlInputSubmit>("confirmButton");
        checkForXss(ActiveBrowser.ViewSourceString());
    }
}
```

Selenium, Web Aii - możliwości

- Możliwość pełnego odwzorowania działań użytkownika
- Klikanie elementów, wybór elementów z listy, przejścia do nowych stron, wywoływanie zdarzeń, możliwość sterowania pracą przeglądarki itd.
- Selenium oraz Web Aii nie są narzędziami wyspecjalizowanymi w testowaniu bezpieczeństwa aplikacji webowych.

Budowa własnego fuzzera - architektura



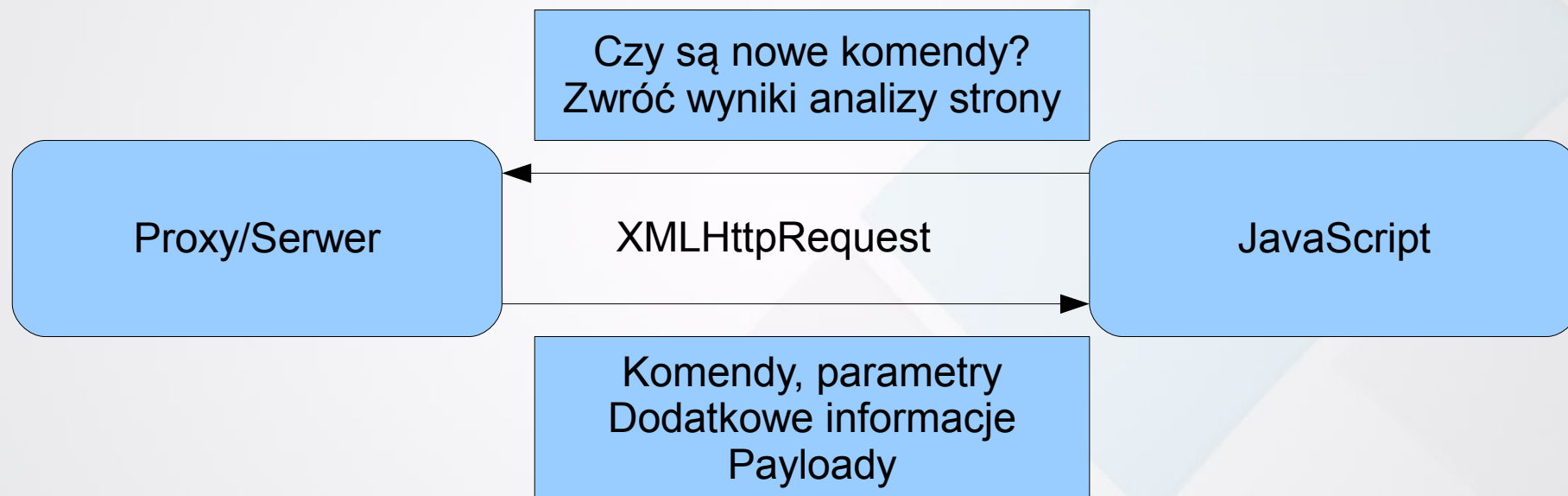
Budowa własnego fuzzera - serwer/proxy

- Pośredniczenie w połączeniu pomiędzy testowaną aplikacją a aplikacją testującą
- Wbudowane metody generowania payloadów
- Analiza nagłówków HTTP
- Przygotowanie scenariuszy testowych na podstawie plików XML
- Analiza źródeł stron i wykrywanie błędów
- Śledzenie nowych stron

Budowa własnego fuzzera - przeglądarka

- Przeglądarka (Firefox) uruchamia się automatycznie.
- Budowany jest tymczasowy profil (prefs.js):
 - `"user_pref("capability.policy.default.Window.alert", "noAccess");"`
 - `"user_pref("capability.policy.default.Window.confirm", "noAccess");"`
 - `"user_pref("capability.policy.default.Window.prompt", "noAccess");"`
 - `"user_pref("network.proxy.http", "localhost");"`
 - `"user_pref("network.proxy.http_port", port);"`

Budowa własnego fuzzera - komunikacja



Budowa własnego fuzzera - javascript

- Wykonywanie poleceń otrzymanych od serwera
- Analiza drzewa DOM
- Symulowanie działań użytkownika
- Analiza treści strony

SE Conference

9 - 10 kwietnia, Kraków

Added!

orderid: "";!--"<alert(XSS)>=&{}
desc: "";!--"<alert(XSS)>=&{}
Logout
Admin

[Logout](#)
[Admin](#)

[elitesolutions.pl](#)
[fuzzing.eu](#)

```
insert
insert
define
change
submit
```

Problemy

- Self Origin Policy
- Walidacja danych po stronie przeglądarki
- Wykrywanie podatności
- Wykrywanie nowo otwartych stron

Live demo

Ograniczenia

- Mniejsza wydajność (request, response, wygenerowanie strony, analiza drzewa DOM...)
- CAPTCHA
- Trudności w implementacji
- Konieczność układania scenariuszy testowych

Pytania

Dziękuję za uwagę!