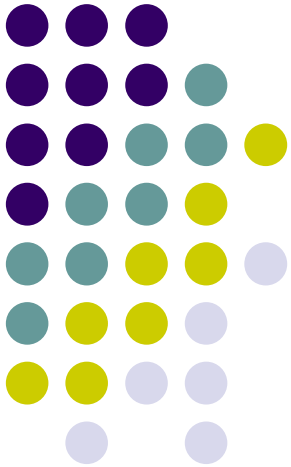
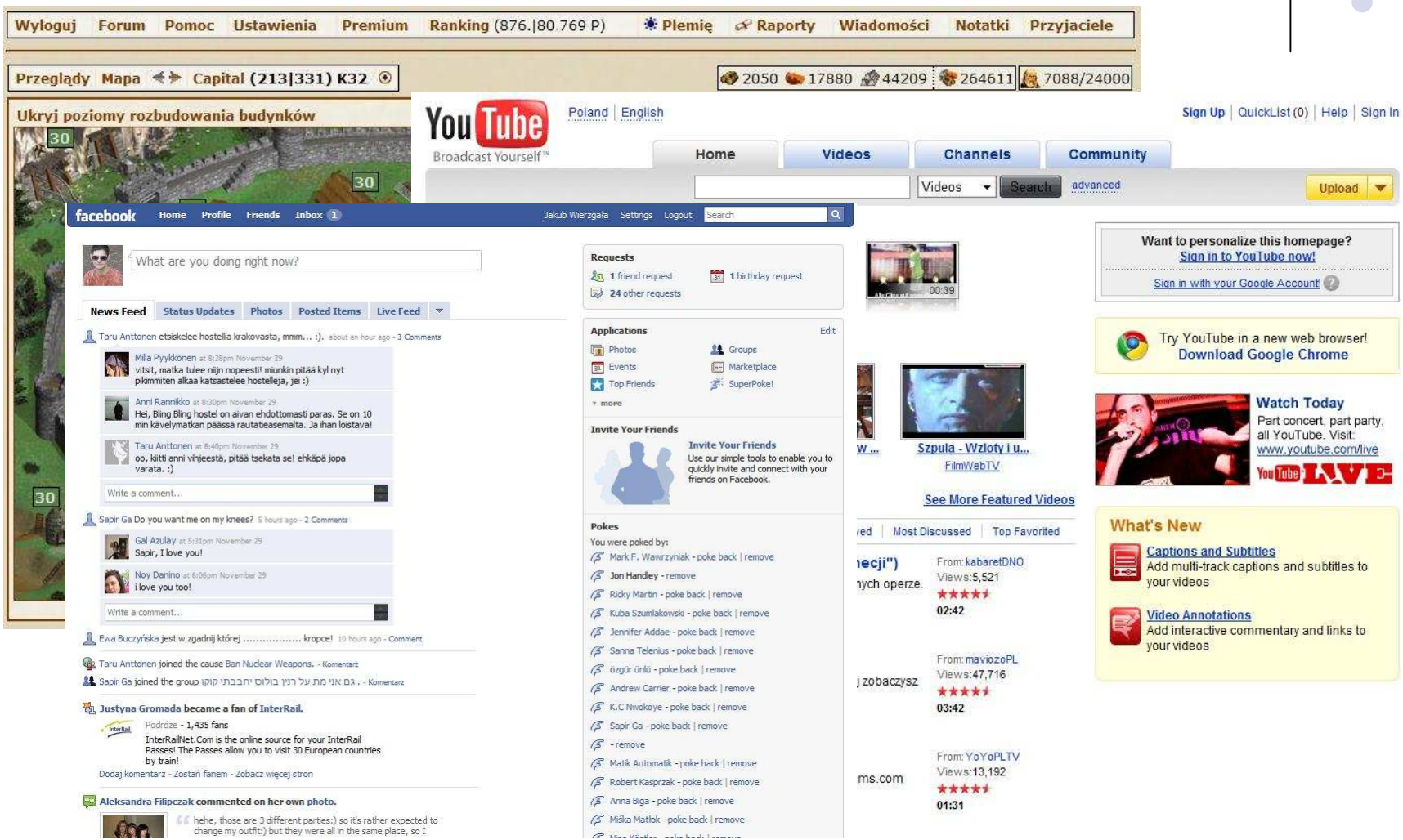
Ajax a bezpieczeństwo aplikacji webowych

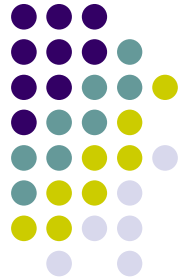
Jakub Wierzgała



Web 2.0


Wyloguj Forum Pomoc Ustawienia Premium Ranking (876.|80.769 P) Plemię Raporty Wiadomości Notatki Przyjaciele
Przeglądy Mapa Capital (213|331) K32
2050 17880 44209 264611 7088/24000
Ukryj poziomy rozbudowania budynków
YouTube Broadcast Yourself
Poland | English
Sign Up | QuickList (0) | Help | Sign In
Home Videos Channels Community
Search advanced Upload
facebook Home Profile Friends Inbox
What are you doing right now?
News Feed Status Updates Photos Posted Items Live Feed
Requests
1 friend request 1 birthday request
24 other requests
Applications
Photos Groups
Events Marketplace
Top Friends SuperPoke!
Invite Your Friends
Pokes
You were poked by:
Want to personalize this homepage?
Sign in to YouTube now!
Sign in with your Google Account!
Try YouTube in a new web browser!
Download Google Chrome
Szpula - Wzloty i u...
FilmWebTV
See More Featured Videos
Most Discussed Top Favorited
Watch Today
Part concert, part party, all YouTube. Visit: www.youtube.com/live
What's New
Captions and Subtitles
Add multi-track captions and subtitles to your videos
Video Annotations
Add interactive commentary and links to your videos

Web 2.0

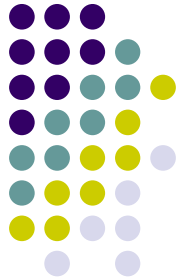


- Zawartość tworzona przez użytkowników
- Wysoka interaktywność
- Aplikacja webowa platformą do działań użytkowników

Web 2.0 - konsekwencje

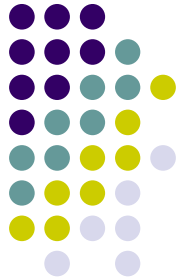
- Brak kontroli nad treścią serwisu przez jego twórcę
- Zwiększenie powierzchni ataku
- Aplikacja webowa platformą do wykonywania ataków na użytkowników

Ajax



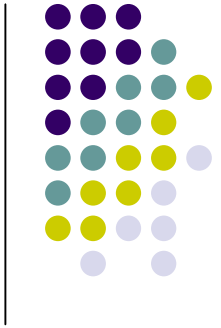
- Nowe podejście do wykorzystania już istniejących technologii
  - XHTML i CSS
  - Document Object Model
  - XML i XSLT
  - XMLHttpRequest
  - JavaScript

Ajax



- Asynchroniczność
- Kod wykonywany i widoczny po stronie klienta
- Komunikacja z serwerem nie tylko na życzenie użytkownika
- Zwiększenie „usability” oraz „responsiveness”

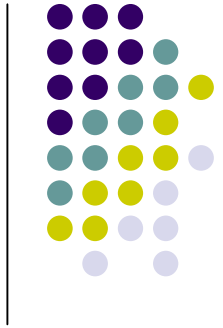
Thick - Thin



Thick-Client	Thin-Client
Logika aplikacji u klienta	Logika aplikacji na serwerze
Potrzeba instalacji	Brak potrzeby instalacji
Mały czas odpowiedzi	Duży czas odpowiedzi
Utrudniona modyfikacja aplikacji	Prosta modyfikacja aplikacji
Przyjazny interfejs	Uproszczony interfejs

A gdzie jest Ajax?

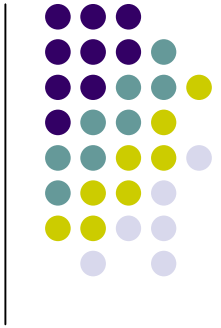
Thick - Thin



Thick-Client	Thin-Client
Logika aplikacji u klienta	Logika aplikacji na serwerze
Potrzeba instalacji	Brak potrzeby instalacji
Mały czas odpowiedzi	Duży czas odpowiedzi
Utrudniona modyfikacja aplikacji	Prosta modyfikacja aplikacji
Przyjazny interfejs	Uproszczony interfejs

Połączenie najbardziej pożądanych cech!!!

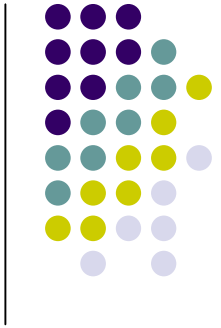
Bezpieczeństwo



Thick-Client	Thin-Client
Logika aplikacji u klienta	Logika aplikacji na serwerze
Wiadomości wysyłane pomiędzy serwerem i klientem trudne do odczytania	Wiadomości wysyłane pomiędzy serwerem i klientem łatwe do odczytania
Aplikacja dostępna tylko dla posiadaczy programu klienckiego	Aplikacja dostępna dla wszystkich

A gdzie jest Ajax?

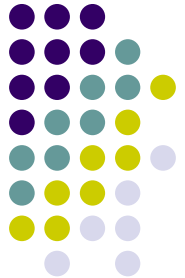
Bezpieczeństwo



Thick-Client	Thin-Client
Logika aplikacji u klienta	Logika aplikacji na serwerze
Wiadomości wysyłane pomiędzy serwerem i klientem trudne do odczytania	Wiadomości wysyłane pomiędzy serwerem i klientem łatwe do odczytania
Aplikacja dostępna tylko dla posiadaczy programu klienckiego	Aplikacja dostępna dla wszystkich

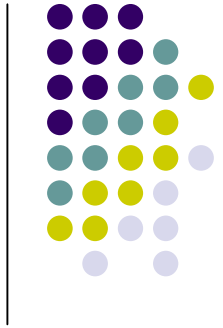
Połączenie najmniej pożądanych cech !!!

Doskonałe połączenie



- Aplikacje Ajax w stosunku do aplikacji „klasycznych”
  - Bardziej złożone
  - Bardziej transparentne
  - Większe
- Skutek: Doskonałe połączenie cech zmniejszających bezpieczeństwo aplikacji

Umiejętności twórców



- Nowa technika
- Brak doświadczenia twórców
- Brak gruntownej wiedzy o działaniu aplikacji

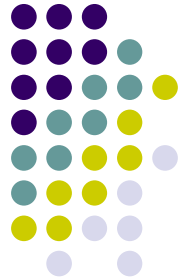
Rzeczywistość

Wysokie wymagania
+
Krótkie terminy
+
Ograniczone możliwości szkolenia
+
Dostęp do frameworków
=
Duża grupa programistów, którzy wiedzą, że aplikacja działa, ale nie wiedzą jak.

Ataki na aplikacje webowe

- Wyliczanie zasobów
- Manipulacja parametrami
  - SQL Injection
  - XPath Injection
  - Cross Site Scripting
  - Przechwytywanie sesji
- Cross Site Request Forgery
- Denial-of-Service

Ataki na aplikacje Ajax

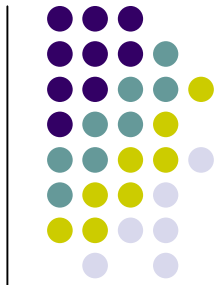


Wszystkie powyższe!!!

,ale

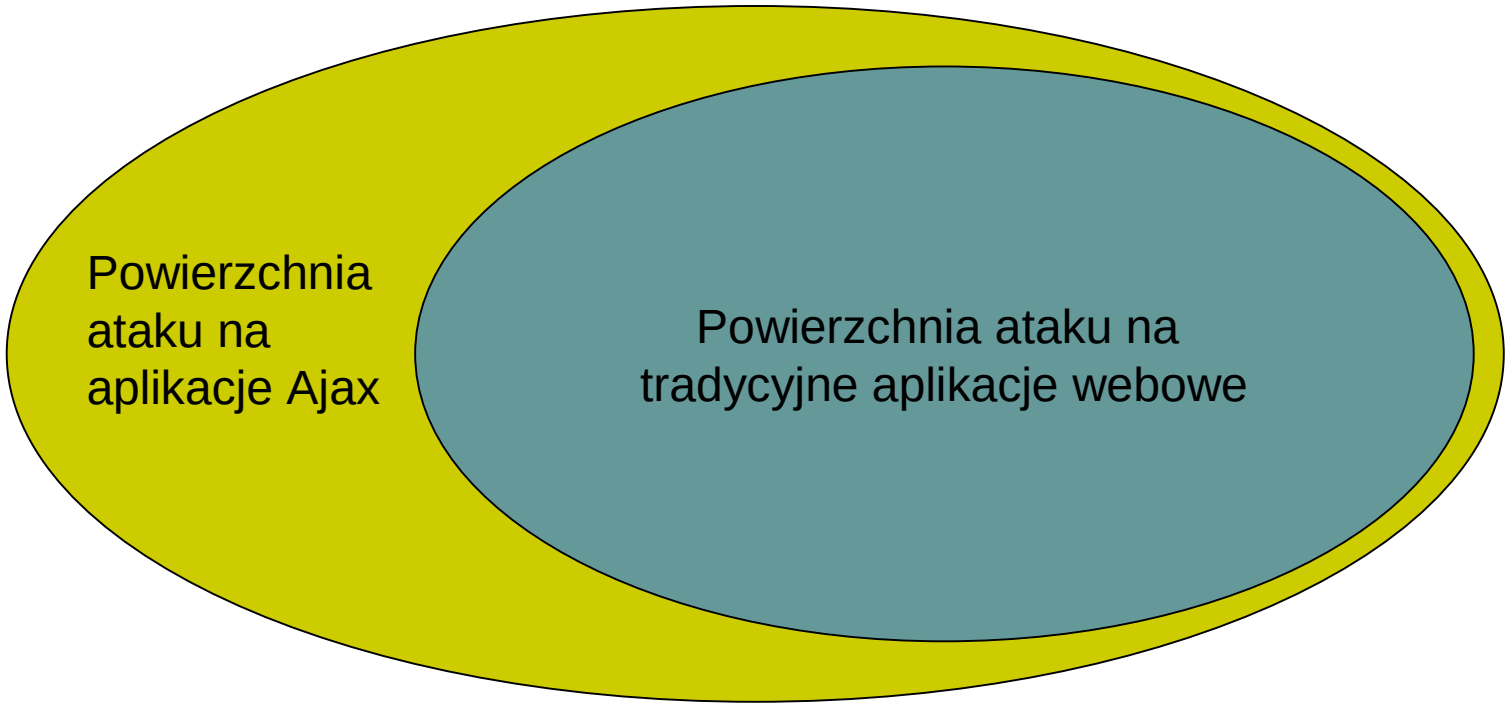
- Zwiększona powierzchnia ataku
- Wgląd w kod aplikacji
- Ataki specyficzne dla Ajax

Powierzchnia ataku na aplikacje Ajax

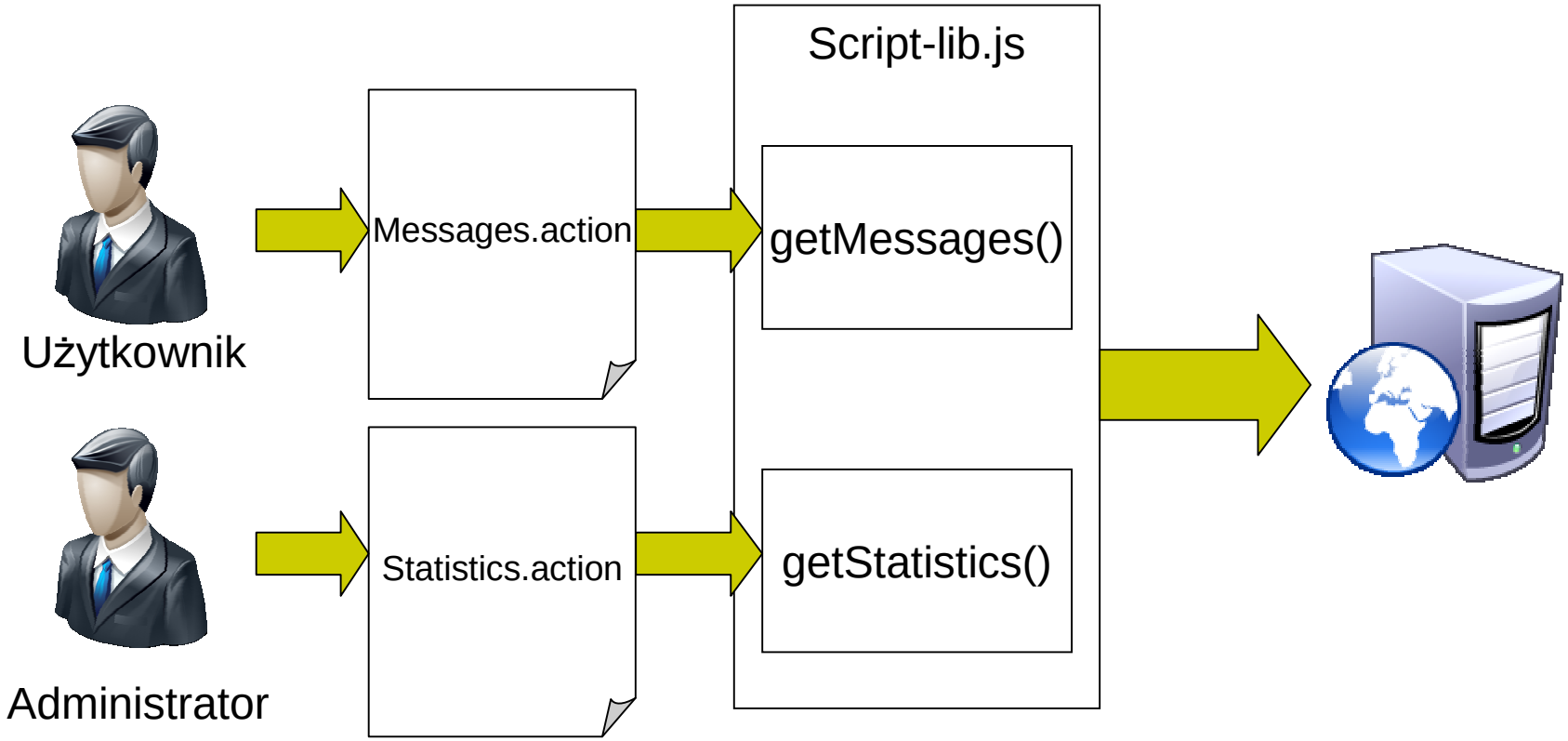


- Pola formularzy
- Cookies
- Nagłówki HTTP
- Ukryte pola formularzy
- Parametry zapytań
- Uploadowane pliki
- Zdalne metody Ajax

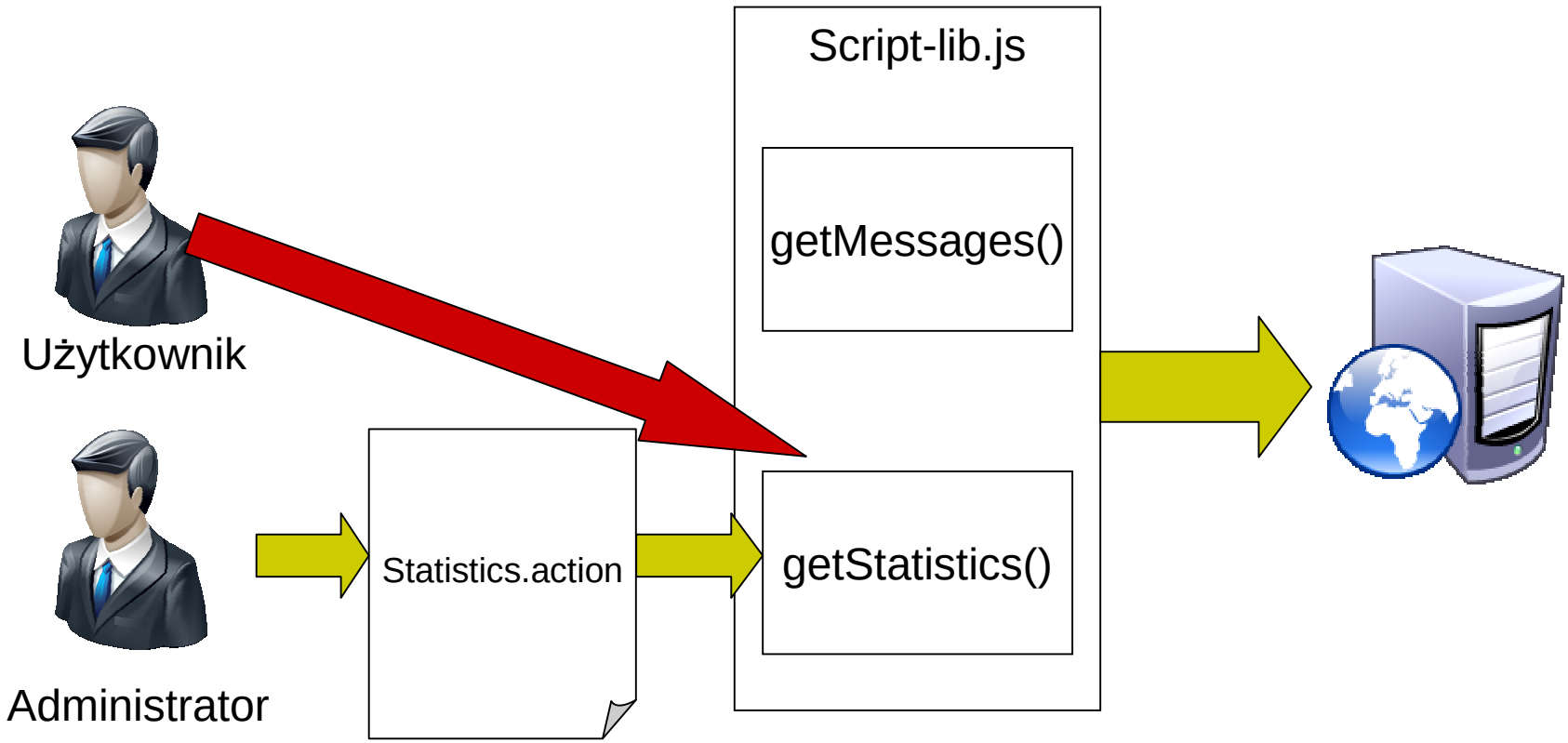
Powierzchnia ataku na aplikacje Ajax



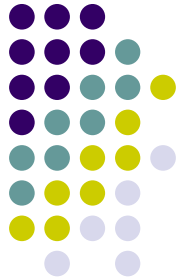
Niebezpieczeństwa Ajax: Transparentność aplikacji



Niebezpieczeństwa Ajax: Transparentność aplikacji



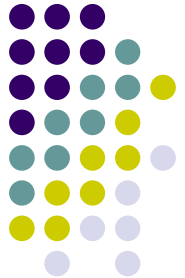
Niebezpieczeństwa Ajax: Transparentność aplikacji



- Stan sesji przechowywany w JavaScript

```
// pobierz cenę piosenki
var songPrice = getSongPrice(songId);
// sprawdź czy użytkownik posiada wystarczająco dużo środków
if (getAccountBalance(username) < songPrice)
{
   alert('Nie posiadasz wystraczjącej ilości środków');
   return;
}
// pobierz środki z konta użytwkonika
debitAccount(username, songPrice);
```

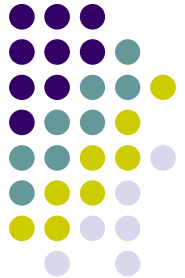
Niebezpieczeństwa Ajax: Transparentność aplikacji



- Ujawnienie danych

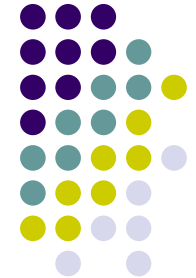
```
<script type="text/javascript">
function processDiscountCode(discountCode) {
   if (discountCode == "HALF-OFF-MUSIC") {
   // przekieruj do strony zamówienia z upustem
      window.location = "SecretDiscountOrderForm.html";
   }
}
</script>
```

Niebezpieczeństwa Ajax: Transparentność aplikacji

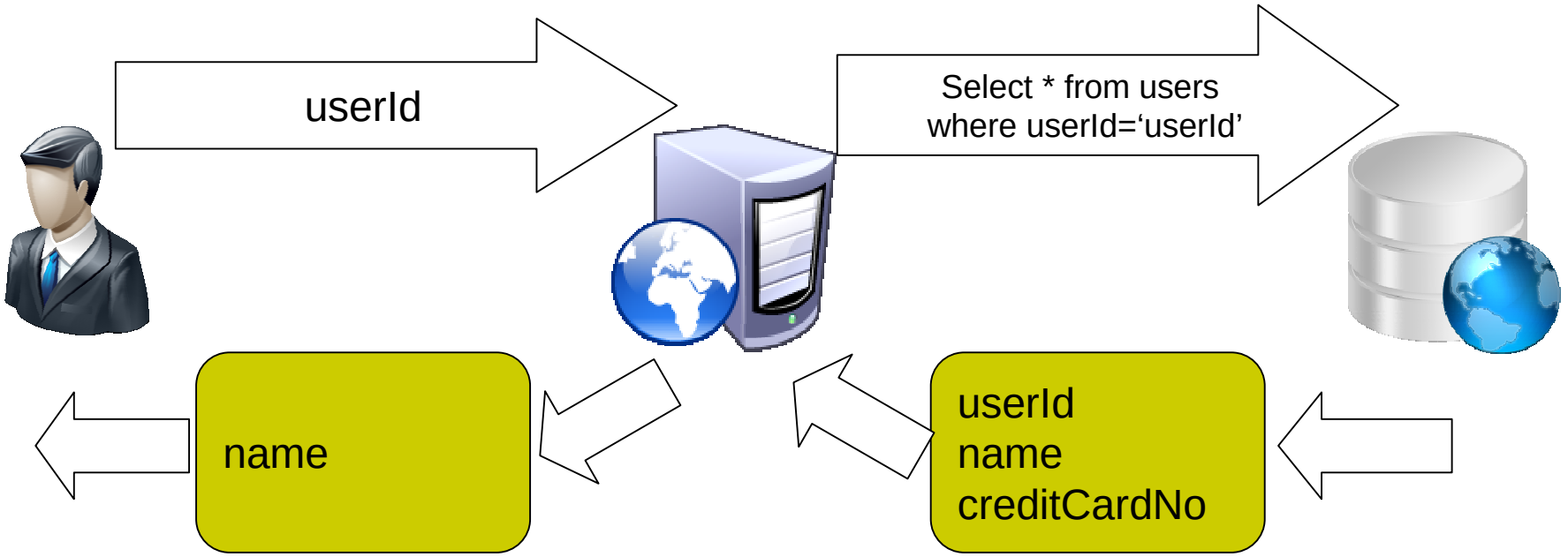


- Komentarze i dokumentacja w kodzie
- Ułatwienie dla atakującego
- Nierzadko kompletny opis działania, parametrów oraz przeznaczenia metod

Niebezpieczeństwa Ajax: Transparentność aplikacji

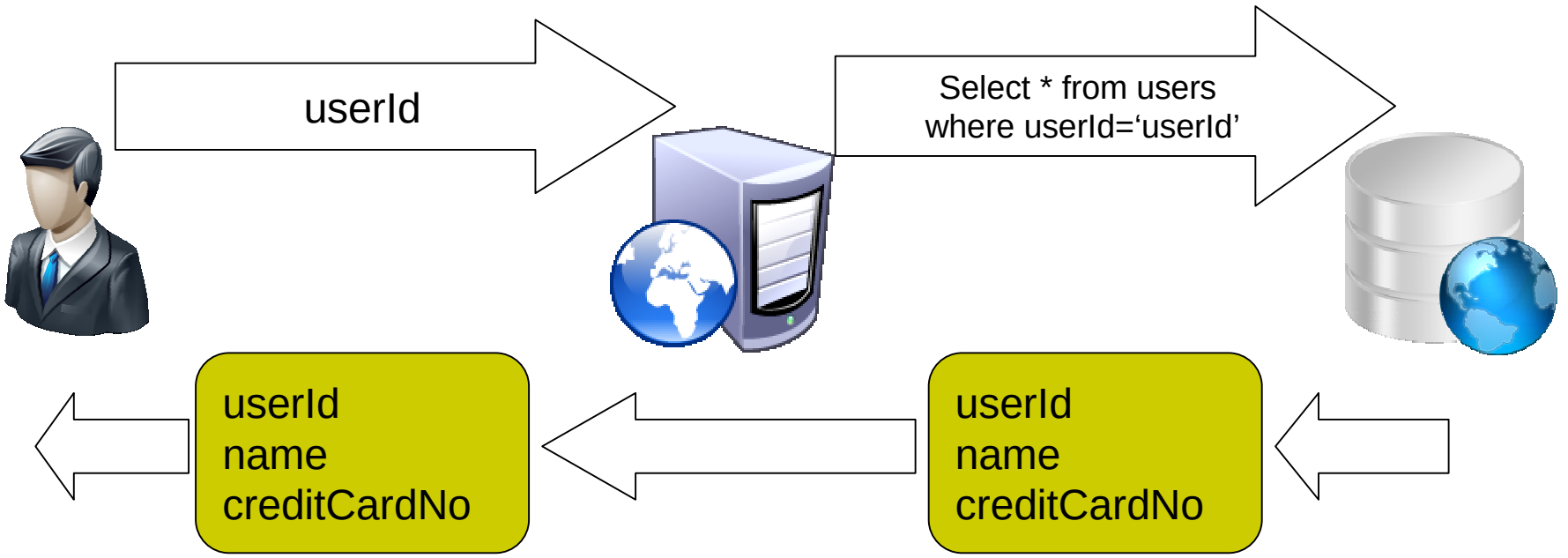


- Manipulowanie danymi po stronie klienta



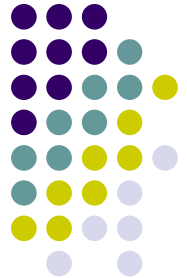
Działanie tradycyjnej aplikacji

Niebezpieczeństwa Ajax: Transparentność aplikacji



Działanie źle zaprojektowanej aplikacji Ajax

Niebezpieczeństwa Ajax: Transparentność aplikacji

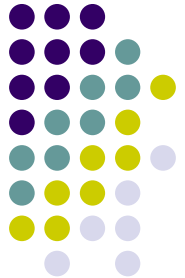


- Środki zaradcze
  - Zaciemnianie kodu
  - Autoryzowanie wywoływania metod po stronie serwera
  - Unikanie zbiorczych bibliotek skryptów
  - Przesyłanie do klienta minimalnego zestawu danych
  - Unikanie komentarzy i dokumentacji w kodzie

Niebezpieczeństwa Ajax: Przesłanianie funkcji

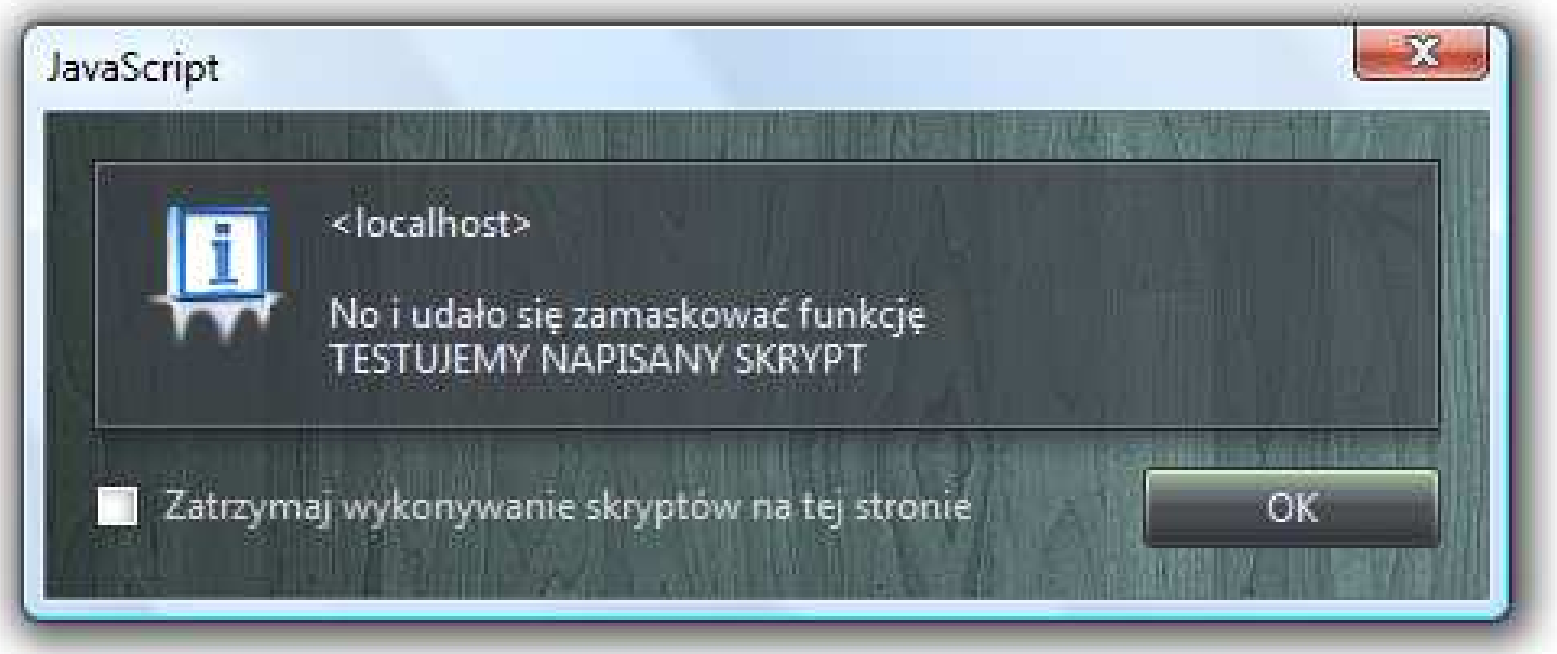
- Typowo w aplikacjach
  - wrażliwych na XSS
  - umożliwiających ładowanie własnego kodu JS (np. JavaScript widget)
  - rodzaju JavaScript mashup
- Bazuje na fakcie, że JavaScript pozwala na przedefiniowanie funkcji (nawet tych wbudowanych)
- Kto ostatni ten lepszy

Niebezpieczeństwa Ajax: Przesłanianie funkcji

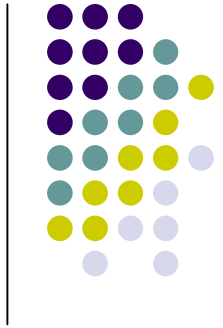


```
<script>
   //tworzymy referencję do funkcji orginalnej
   var oldAlert = window.alert;
   //tworzymy funkcje maskującą
   function newAlert(msg)
   {
      out = "No i udało się zamaskować funkcję\n";
      out +=msg.toUpperCase();
      oldAlert(out);
   }
   //zmieniamy referencje funkcji
   window.alert = newAlert;
   alert("Testujemy napisany skrypt");
</script>
```

Niebezpieczeństwa Ajax: Przesłanianie funkcji



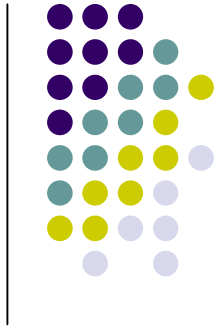
Niebezpieczeństwa Ajax: Przesłanianie funkcji



Możliwości???

Ograniczone niemal jedynie wyobraźnią użytkownika!

Same-Origin Policy



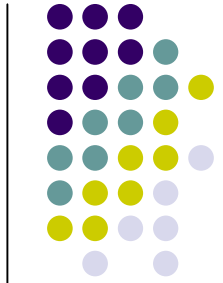
- Kod JavaScript może uzyskać dostęp tylko do dokumentów pochodzących z tej samej domeny co on sam.
- Reguła implementowana przez wszystkie wiodące przeglądarki

JSON

- Notacja pozwalająca na przedstawianie tablic i obiektów w JavaScript
- Definicja tablicy jest interpretowana jako poprawny kod JavaScript; wywołanie konstruktora Array()

```
[["Jakub", "Wierzgała", "wierzgala@gmail.com"],
["Jan", "Kowalski" kowalski@foobar.com]],
```

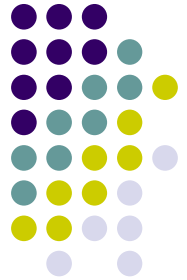
JSON Hijacking



- Połączenie ataku CSRF oraz przesłaniania funkcji JavaScript.
- CSRF umożliwia zmuszenie atakowanego klienta do wysłania dowolnego żądania HTTP, ale nie pozwala na podejrzenie odpowiedzi przez atakującego

Z pomocą przychodzi przesłanianie funkcji

JSON Hijacking



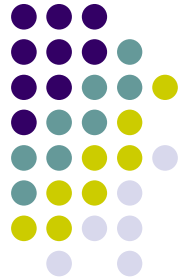
- Scenariusz:
  - Atakujący tworzy złośliwą stronę
  - Nadpisuje funkcję konstruującą tabele JSON
  - Na stronie zamieszcza żądanie HTTP do atakowanego serwisu
  - Nakłania atakowanego do wejścia na utworzoną stronę

JSON Hijacking

- Przykład nadpisania konstruktora tabeli

```
function Array()
{
   var foo = this;
   var bar = function()
   {
      var ret = „Przechwycona tablica: [";
      for(var x in foo) {
         ret += foo[x] + ", ";
      }
      ret += "]";
      //powiadom atakującego (w tym przypadku tylko wyświetlamy)
      alert(ret);
   };
   setTimeout(bar, 100);
}
```

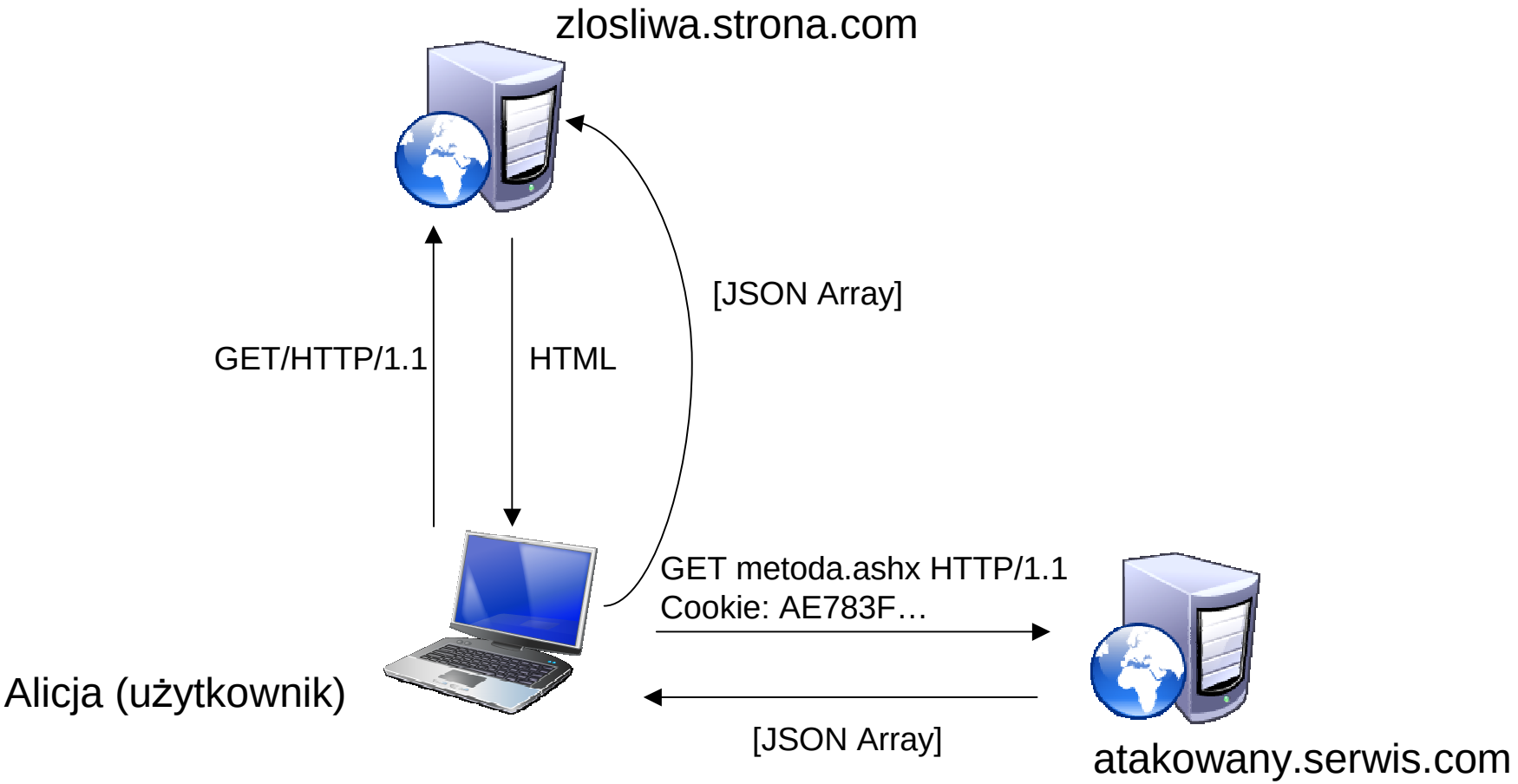
JSON Hijacking



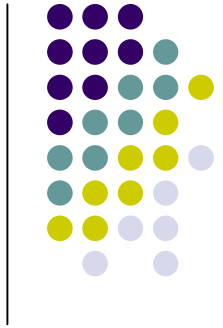
- Zawartość złośliwej strony

```
<html>
   <head>
      <title>JSON Hijacking Demo</title>
   </head>
   <body>
      <script>
         function Array() {<!--wcześniej pokazany kod-->}
      </script>
<!–-ładowanie skryptu bezpośrednio z atakowanego serwisu-->
      <script src=
      "http://www.atakowany.serwis.com/Funkcje/ajaxcalls/" +
      „metoda.ashx">
      </script>
   </body>
</html>
```

JSON Hijacking

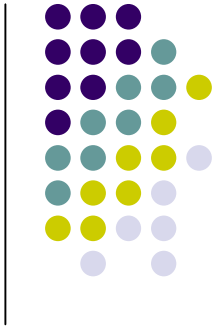


JSON Hijacking



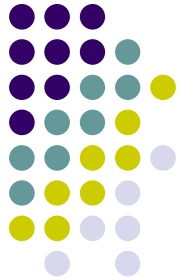
- Jak się zabezpieczyć?
  - Dodanie niepoprawnej linii kodu przed definicją tablicy (np. **`I'/\/\ a bl0ck of inva1id $ynT4x!)`**
  - Zamknięcie definicji tablicy wewnątrz komentarza
  - Dodanie nieskończonej pętli przed definicją tablicy
    - for(;;)
    - while(1)

JSON Hijacking



- Jak się zabezpieczyć?
  - Dodanie niepoprawnej linii kodu przed definicją tablicy (np. **`I'/\/\ a bl0ck of inva1id $ynT4x!)`**
  - Zamknięcie definicji tablicy wewnątrz komentarza
  - Dodanie nieskończonej pętli przed definicją tablicy
    - for(;;)
    - while(1)

Ataki na aplikacje Ajax



- Wykradanie danych przechowywanych po stronie klienta
- Atakowanie warstwy prezentacji
- Atakowanie serwisów typu „mashup”
- JavaScript Worms
- … i wiele innych

Podsumowanie

- Użycie Ajax zmniejsza bezpieczeństwo aplikacji
- Niski poziom wiedzy twórców na temat działania aplikacji
- Niski poziom świadomości zagrożeń
- Zwiększenie powierzchni „starych” ataków
- Nowe rodzaje ataków specyficzne tylko dla Ajax