

Sicherheit und der Ajax-Client

Carsten Eilers

Vorstellung

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



Agenda

- Vorbemerkungen
- Klassiker: XSS und CSRF
- SQL-Injection
- Presentation Layer
- Clickjacking
- „Dies und Das“

Vorbemerkungen

Zuerst die Klassiker - alt, aber bewährt

Danach die Newcomer - weniger bekannt und oft unterschätzt

Agenda

- Vorbemerkungen
- **Klassiker: XSS und CSRF**
- SQL-Injection
- Presentation Layer
- Clickjacking
- „Dies und Das“

Ein Klassiker: XSS (1)

„Same-Origin-Policy“:

Von einer vertrauenswürdigen Webseite mitgelieferter JavaScript-Code ist vertrauenswürdig

Durch XSS verletzt

Ein Klassiker: XSS (2)

3 Arten von XSS:

- DOM-basiertes XSS
Einschleusen über präparierte URL
- Reflektiertes XSS
Einschleusen über präparierte URL oder Formular
- Persistentes XSS
Einschleusen über z.B. Gästebuch

Ein Klassiker: XSS (3)

Folgen:

- Wer JavaScript ausführen kann, kontrolliert den Browser
- Wer den Browser kontrolliert, ist hinter der Firewall
- Wer den Browser kontrolliert, ist nur einen Schritt von der Kontrolle über das System entfernt

Egal?

Langweilig?

Wieso diese Wiederholung?

Warum sollte jemand das System übernehmen, wenn der Browser für seine Zwecke reicht?

Port 80 ist fast überall offen, mit WebSockets wird das Loch in der Firewall noch größer

Vorausschauend handeln

Heute programmieren Sie eine Webanwendung, die in einem Browser läuft

Demnächst programmieren Sie eine Netzwerk-Anwendung, die in einem eigenen System (dem Browser) läuft

Die Verantwortung steigt - sind Sie bereit?

Zweiter Klassiker: CSRF

- Authentifizierung nicht für jede Aktion, sondern Speicherung des Benutzerstatus (z.B. Cookie, HTTP-Basic-Authentication)
- Status wird vom Browser automatisch mit jedem Request mitgeschickt
- Auch, wenn nicht der Benutzer, sondern ein Skript den Request auslöst

Gegenmaßnahmen (1)

Vertraue nie dem Client!

**Prüfungen auf dem Client für den Client
(Bequemlichkeit des Benutzers)**

**Prüfungen auf dem Server für den Server
(Sicherheit des Servers)**

Gegenmaßnahmen (2)

XSS:

Eingaben filtern

CSRF:

**nicht vorhersagbare Token in jedem Formular,
nur Request mit Token ausführen**

zusätzlich erneutes Login und/oder CAPTCHA

Agenda

- Vorbemerkungen
- Klassiker: XSS und CSRF
- **SQL-Injection**
- Presentation Layer
- Clickjacking
- „Dies und Das“

SQL-Injection im Client

SQL-Injection im Client?

Prinzipiell ja, wenn die Browser erst mal eine eigene SQL-Datenbank als Clientseitigen Speicher bekommen

Hier und jetzt aber etwas anders gemeint

SQL-Injection klassisch (1)

Beispiel:

Webservice mit SQL-Injection-Schwachstelle

```
SELECT * FROM Benutzer WHERE ID = $benutzerID
```

Angriff normalerweise:

```
123 UNION SELECT * FROM Kreditkarten
```

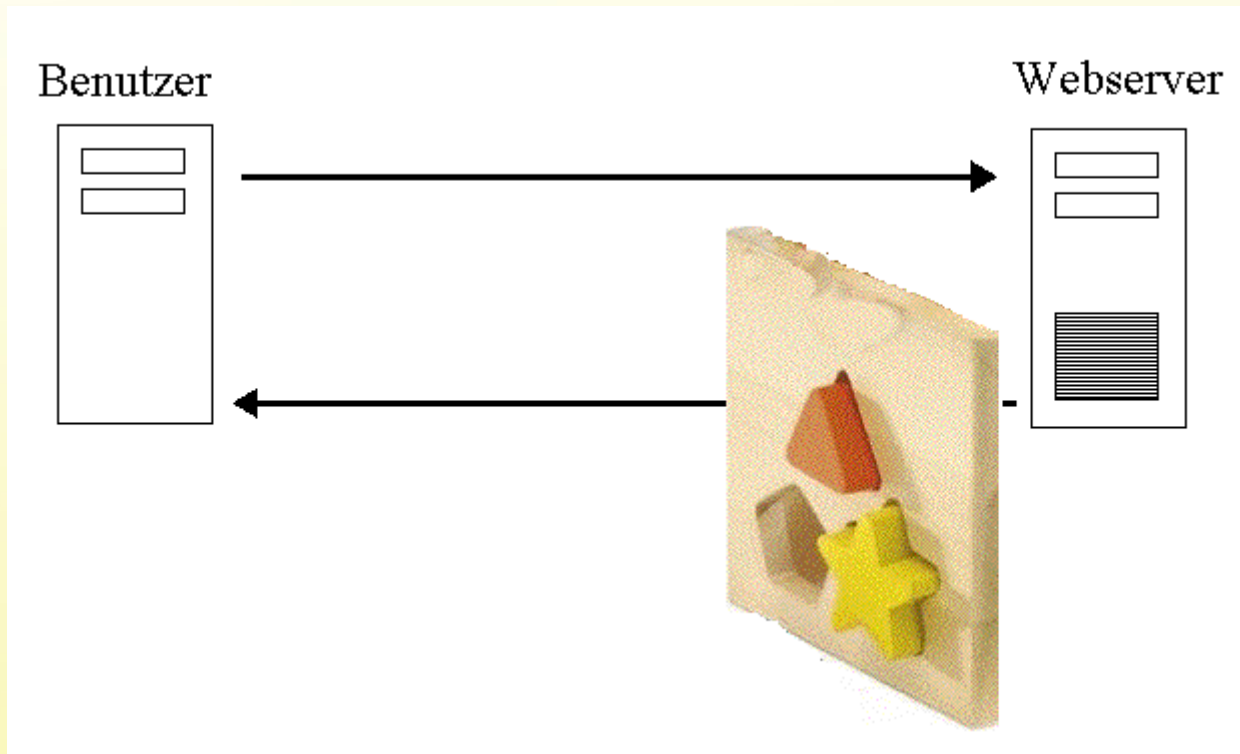
SQL-Injection klassisch (2)

Webanwendung wandelt Daten um, muss sie also verarbeiten können:

- identische Spaltenanzahl
- gleiche oder umwandelbare Datentypen

Voraussetzungen nicht erfüllt: Keine Ausgabe an den Client

SQL-Injection klassisch (3)



SQL-Injection via Client (1)

Ajax-Client wandelt die Ausgabe um:

Angriff kommt ohne UNION aus:

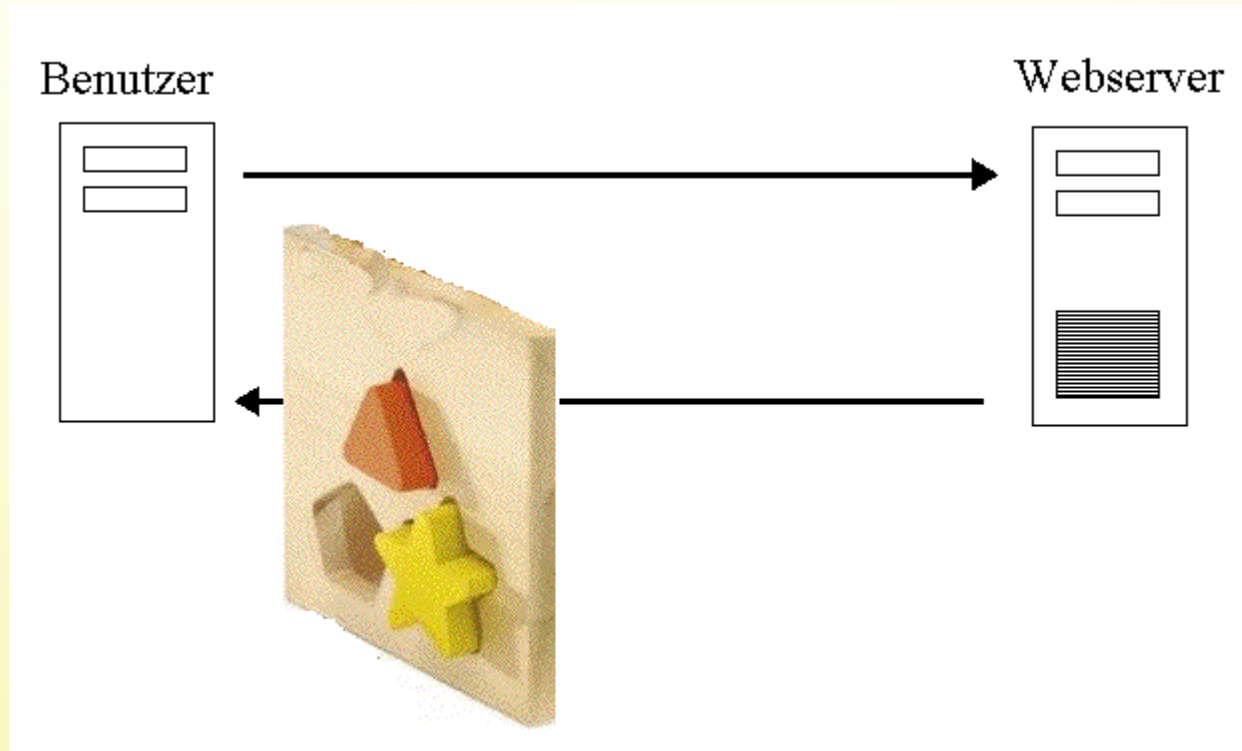
```
SELECT * FROM Benutzer WHERE BenutzerID = 123;
```

```
SELECT * FROM Kreditkarten
```

Server sendet z.B. XML-Daten,

- die der Client nicht darstellen kann,
- die der Angreifer aber über Proxy abfängt

SQL-Injection via Client (2)



SQL-Injection verhindern

Gegenmaßnahmen:

- keine SQL-Injection-Schwachstelle haben (klar)
- Nicht nur Eingaben filtern, sondern auch Ausgaben

Agenda

- Vorbemerkungen
- Klassiker: XSS und CSRF
- SQL-Injection
- **Presentation Layer**
- Clickjacking
- „Dies und Das“

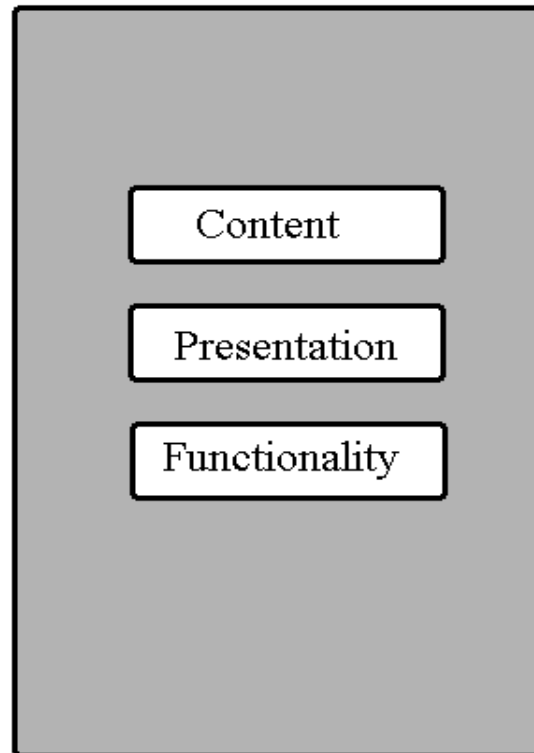
Webseiten (1)

Webseiten enthalten 3 Arten von Informationen:

- Content - Die anzuzeigenden Daten
- Presentation - Anweisungen für die Darstellung
- Functionality - Anweisungen für das Verhalten

Webseiten (2)

seite.html

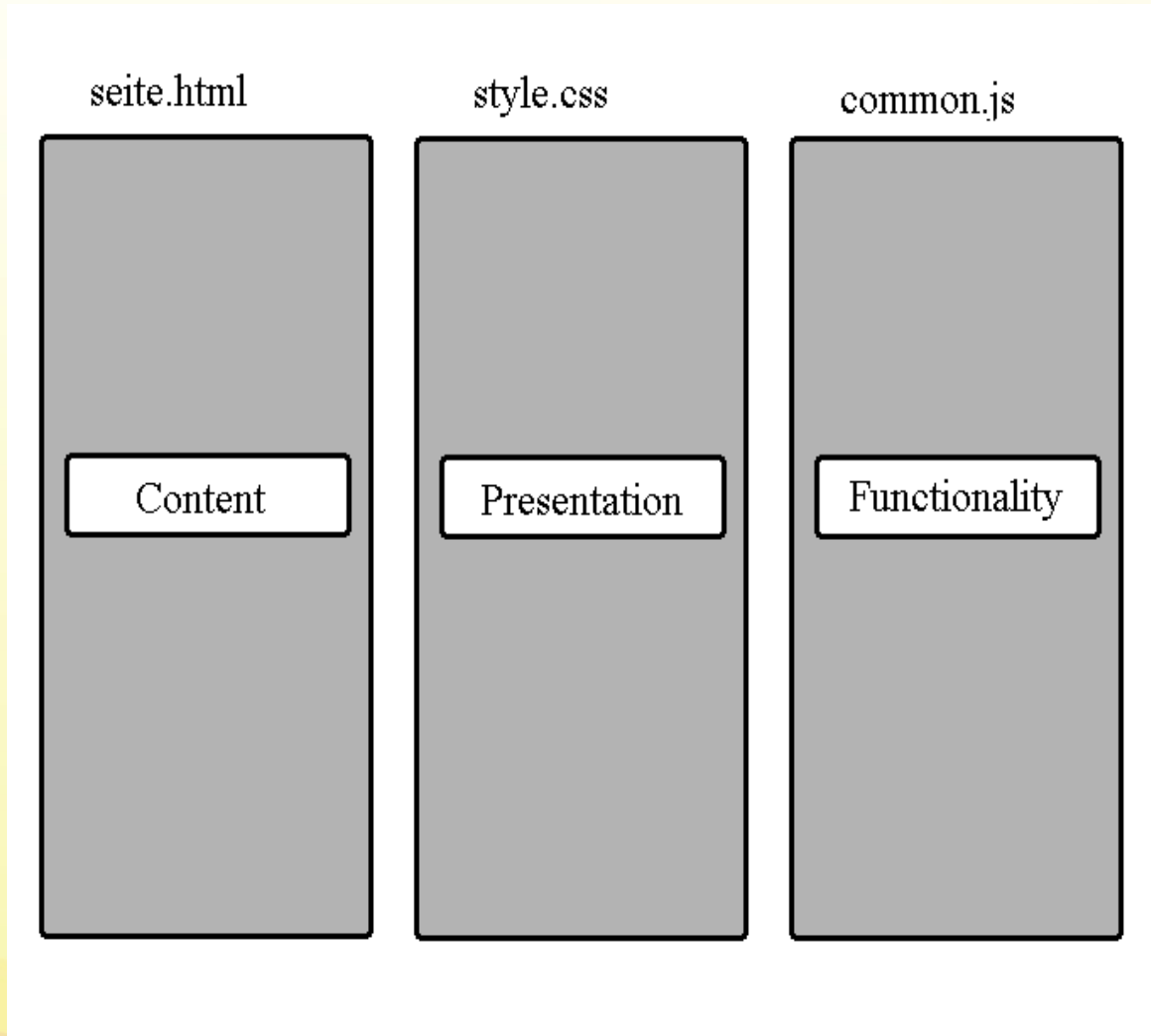


Webseiten (3)

Oft auf vielen Seiten die gleichen Teile

Besser: Bereiche in eigene Dateien aufteilen und verlinken

Webseiten (4)



Webseiten (5)

Praktisch, aber unter Umständen unsicher:

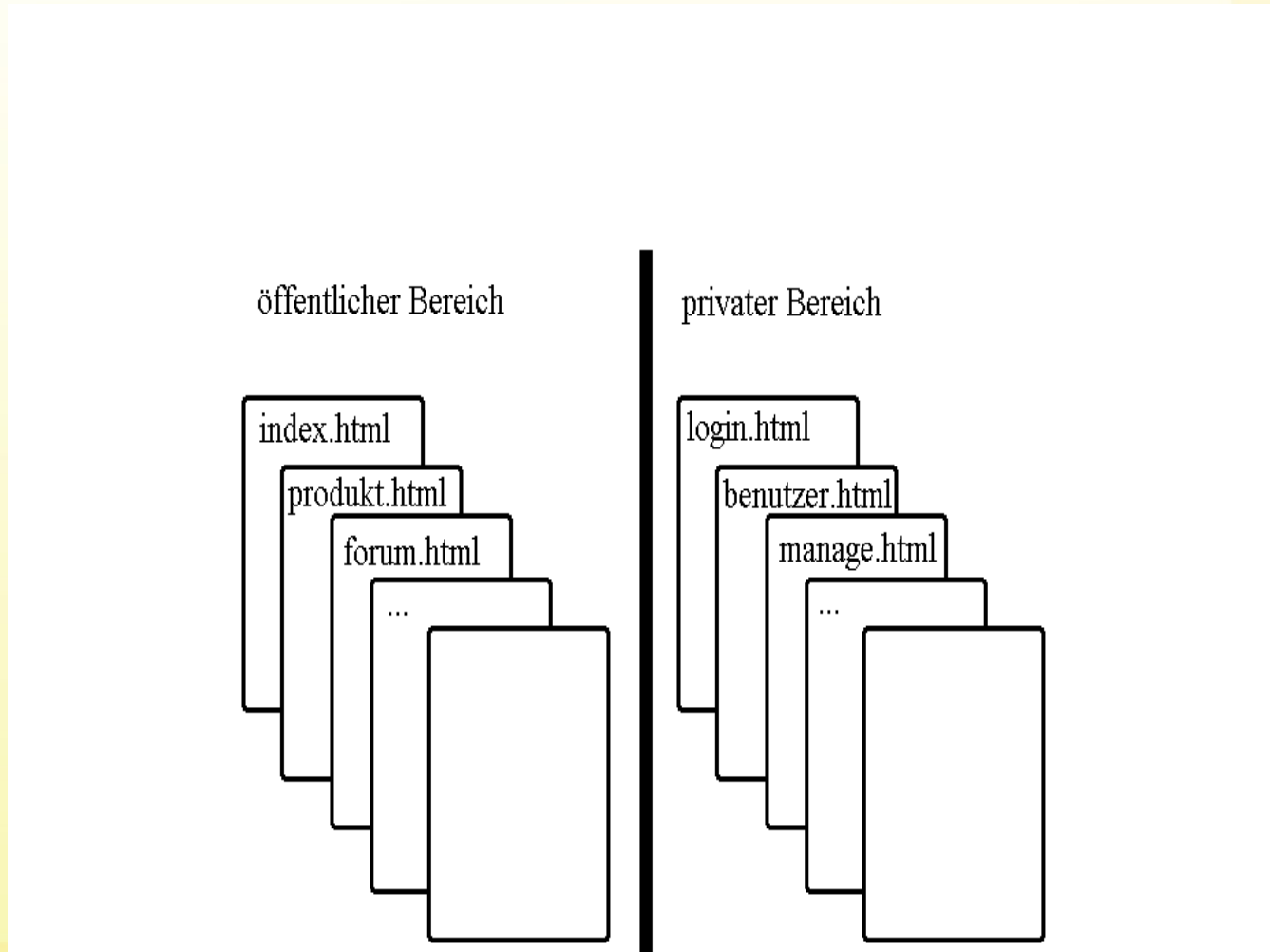
- Preisgabe privater Informationen
- Wenige Dateien bestimmten Aussehen der gesamten Website - auch bei einer Manipulation

Data Mining in CSS (1)

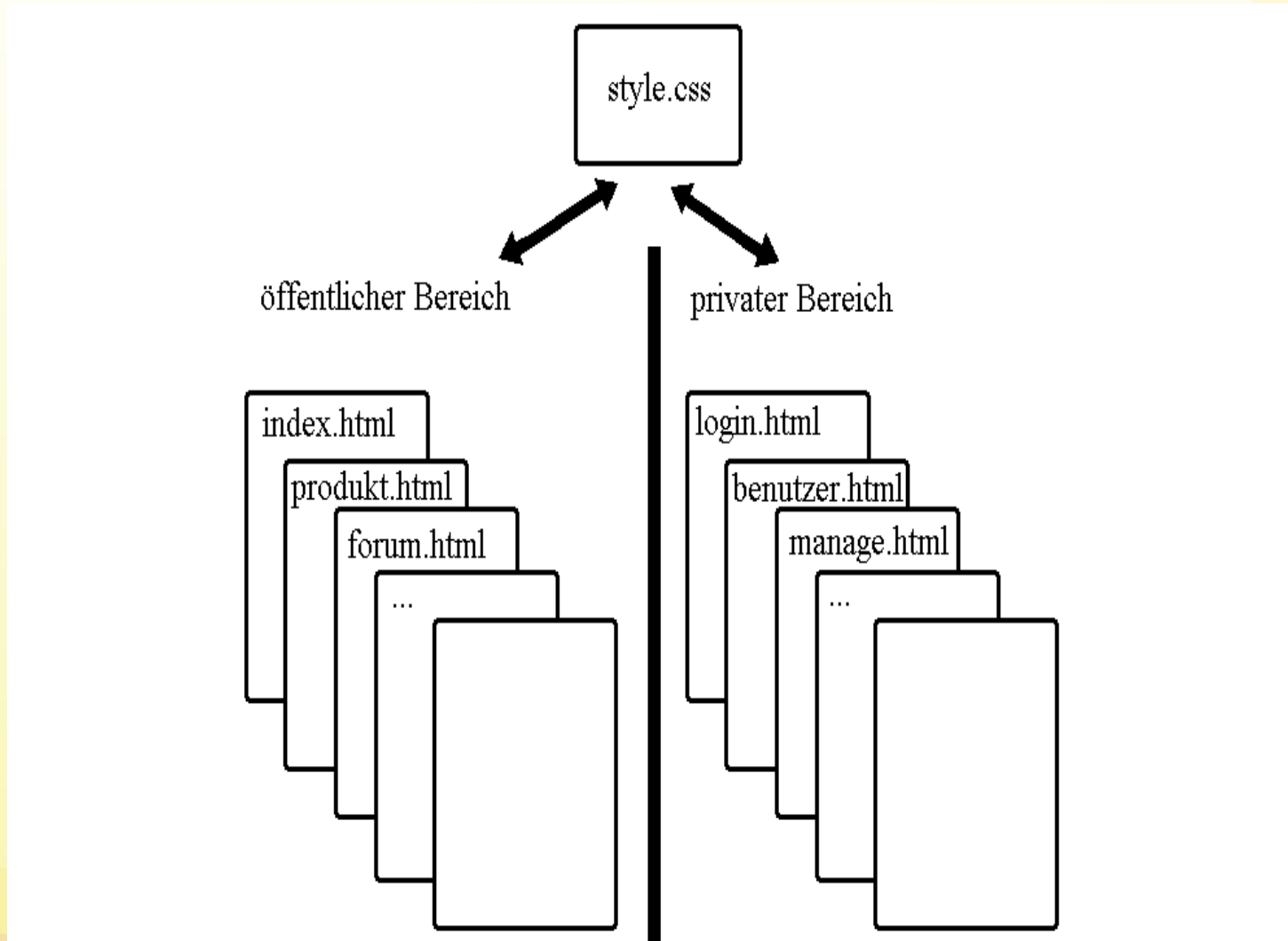
Ein Stylesheet für alle Seiten ...

... einschließlich privater Bereiche, deren Existenz nicht öffentlich bekannt sein soll

Data Mining in CSS (2)



Data Mining in CSS (3)



Data Mining in CSS (4)

Suche nach definierten, aber ungenutzten Styles

Enthaltene URL, z.B. zu einem Bild in einem privaten Bereich, zeigt Angreifer den Weg

Look and Feel Hack

- Oft unterschätzt
- Was kann ein Angreifer schon ausrichten, wenn er eine Datei manipulieren kann?

David Copperfield lässt Eisenbahnwagons verschwinden!

MySpace-Phishing (1)

Gutes Beispiel: Phishing, z.B. bei MySpace

Menüleiste oberhalb fast jeder Seite

Menü befindet sich in div-Tag

Tags lassen sich überschreiben

MySpace-Phishing (2)

```
<style>
```

```
  h1 { color: red; font-family: sans-serif;  
    font-size: 42pt; }
```

```
</style>
```

```
<h1>Testüberschrift 1</h1>
```

```
<style>
```

```
  h1 { color: blue; font-family: courier; font-  
    size: 8pt; }
```

```
</style>
```

```
<h1>Testüberschrift 2</h1>
```

MySpace-Phishing (3)

Was erscheint im Webbrowser?

Weiter mit MySpace:

Benutzer dürfen Style ändern

Phisher überschreiben Style des div-Tags der
Menüleiste

z.B. `display=none`

MySpace-Phishing (4)

Originale Menüleiste wird nicht mehr angezeigt

Gefälschte Menüleiste mit Links zu Phishing-Seiten wird über absolute Positionen an Stelle des jetzt unsichtbaren Originalmenüs gelegt

Bsp: Wurm MySpace-Zango Quicktime
(Dezember 2006)

Erweiterter Angriff (1)

Erweiterter Angriff:

Ajax-Anwendung zum Verkauf von Gütern

- Drop-Down-Menü mit verfügbaren Gütern
- Textbox für die Menge
- „Kaufen“- / „Verkaufen“-Knöpfe für Aktion

Erweiterter Angriff (2)

Knöpfe sind input-Tags mit style-Information:

```
<input type="submit" class="kaufknopf" value="" onclick="klickKaufen();" />
```

```
<input type="submit" class="verkaufknopf" value="" onclick="klickVerkaufen();" />
```

Erweiterter Angriff (3)

```
.kaufknopf {  
    border: 0px;  
    overflow: hidden;  
    background-image: url("kaufen.gif")M  
    background-repeat: no-repeat;  
    width: 110px  
    height: 35px  
}
```

Erweiterter Angriff (4)

```
verkaufknopf {  
    border: 0px;  
    overflow: hidden;  
    background-image: url("verkaufen.gif")M  
    background-repeat: no-repeat;  
    width: 110px  
    height: 35px  
}
```

Erweiterter Angriff (5)

Knöpfe enthalten unterschiedliche Hintergrundbilder und rufen unterschiedliche JavaScript-Funktionen auf

Erweiterter Angriff (6)

Angriff:

Style-Sheet so ändern, dass der Kaufen-Knopf wie der Verkaufen-Knopf aussieht und umgekehrt

Durch absolute Positionierung den Kaufen-Knopf an Stelle des Verkaufen-Knopf setzen und umgekehrt

Erweiterter Angriff (7)

```
.kaufknopf {  
    position: absolute; top 130px; left 130px;  
    border: 0px;  
    overflow: hidden;  
    background-image: url("verkaufen.gif")M  
    background-repeat: no-repeat;  
    width: 110px  
    height: 35px  
}
```

Erweiterter Angriff (8)

```
.verkaufknopf {  
    position: absolute; top 130px; left 5px;  
    border: 0px;  
    overflow: hidden;  
    background-image: url("kaufen.gif")M  
    background-repeat: no-repeat;  
    width: 110px  
    height: 35px  
}
```

Erweiterter Angriff (9)

Nach dem Angriff:

Knopf mit Aufschrift „Kaufen“ an der richtigen Stelle der Webseite - aber es ist das `input`-Tag fürs Verkaufen

Klick auf Kaufen ruft `klickVerkaufen()` auf

Die Webseite ist unverändert, der HTML-Code ist der gleiche wie zuvor!

XSS über CSS (1)

CSS enthalten Präsentations-Informationen,
trotzdem kann JavaScript-Code drin sein:

```
table {  
    background-image:  
    url("javascript:alert('Angriff!');")  
    background-repeat: no-repeat;  
}
```

XSS über CSS (2)

Jede Seite, die das Stile-Sheet einbindet und ein `table`-Tag enthält, enthält den XSS-Code

Auch z.B. mit `body`-Tag möglich:

```
body {  
    background-image:  
    url ("javascript:alert ('Angriff!')");  
    background-repeat: no-repeat;  
}
```

Und wie wird manipuliert?

Betrachten der Stylesheets trivial

Manipulation

- über erlaubte eigene Style-Informationen einfach
- über andere Schwachstellen, z.B. Remote File Inclusion oder Code Execution, die Zugriff auf Dateien erlauben
- über Cache-Poisoning-Angriffe

Gegenmaßnahmen

Data Mining: Keine vertrauliche Informationen in öffentliche Dateien, auch nicht, wenn es „nur“ Präsentations-Informationen sind

Eigene StyleSheets müssen geprüft werden (schwierig!)

Agenda

- Vorbemerkungen
- Klassiker: XSS und CSRF
- SQL-Injection
- Presentation Layer
- **Clickjacking**
- „Dies und Das“

Clickjacking

Clickjacking

Wer von Ihnen hat den Begriff schon mal gehört?

Oder 'UI redress attack'?

Clickjacking History (1)

Anfang September:

Jeremiah Grossman und Robert "RSnake" Hansen ziehen Vortrag für OWASP NYC AppSec zurück

Begründung: Vorgestellte Schwachstelle ist so gefährlich, das Hersteller Zeit zum Patchen brauchen

Clickjacking History (2)

„Clickjacking ermöglicht es einem Angreifer, einen Benutzer zum Klick auf etwas gar nicht oder nur kurz sichtbares zu bewegen.“

Clickjacking History (3)

Anfang Oktober:

Guy Aharonovsky veröffentlicht eine Demo, bei der Benutzer eigentlich ein Spiel spielen, aber tatsächlich die Flash-Konfiguration ändern und Mikrofon und Kamera einschalten

Adobe veröffentlicht einen Workaround

Clickjacking History (4)

RSnake veröffentlicht weitere Details:

- weitverbreitetes Problem
- auch ohne JavaScript möglich
- auch ActiveX angreifbar
- Beobachtung von Clicks auf anderen Websites erlaubt Angriffe mit mehreren Schritten

Clickjacking History (5)

Mitte Oktober:

Jeremiah Grossman und Rsnake veröffentlichen ein Paper, auf dem die folgende Beschreibung basiert

Clickjacking praktisch (1)

Voraussetzung:

Das Opfer wird auf eine Seite gelockt, die unter der Kontrolle des Angreifers steht

Angreifer muss mehrere Seiten rendern und JavaScript ausführen können

Clickjacking praktisch (2)

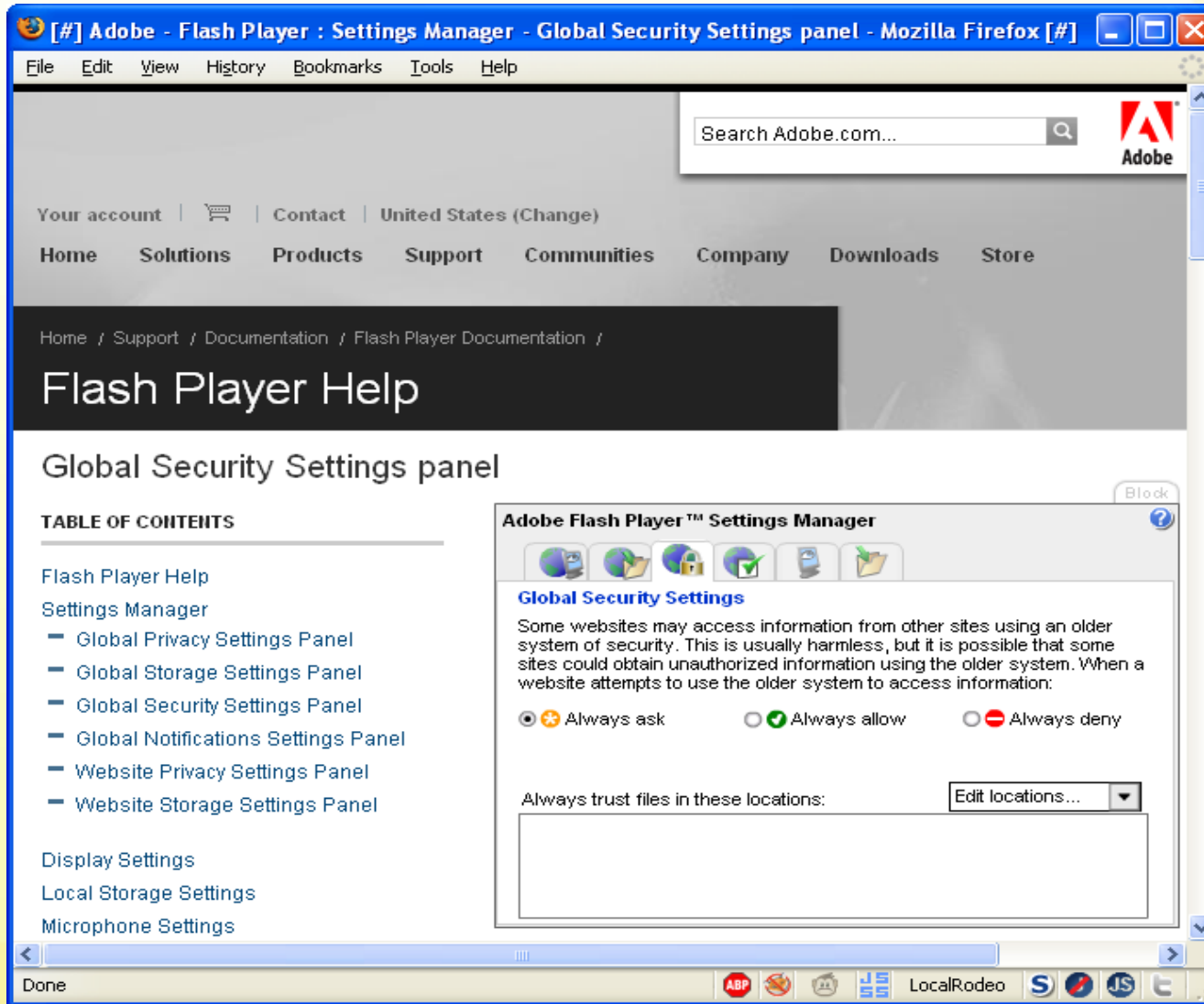
Beispiel:

Adobes Global Settings Manager mit den Sicherheitseinstellungen für den Flash-Player

Vorteil:

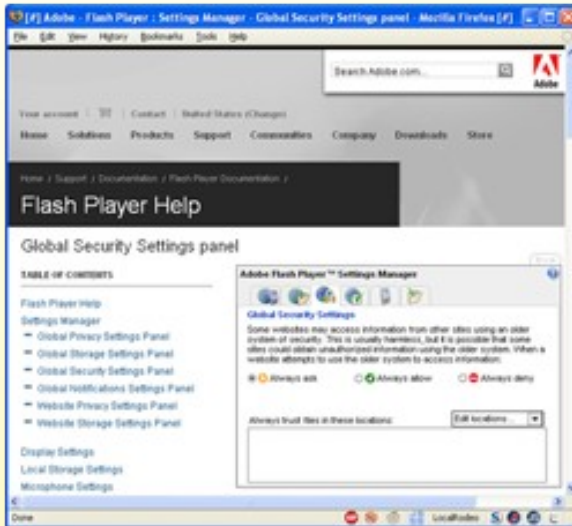
Keine Authentifizierung notwendig, URL ist immer gleich

Clickjacking praktisch (3)



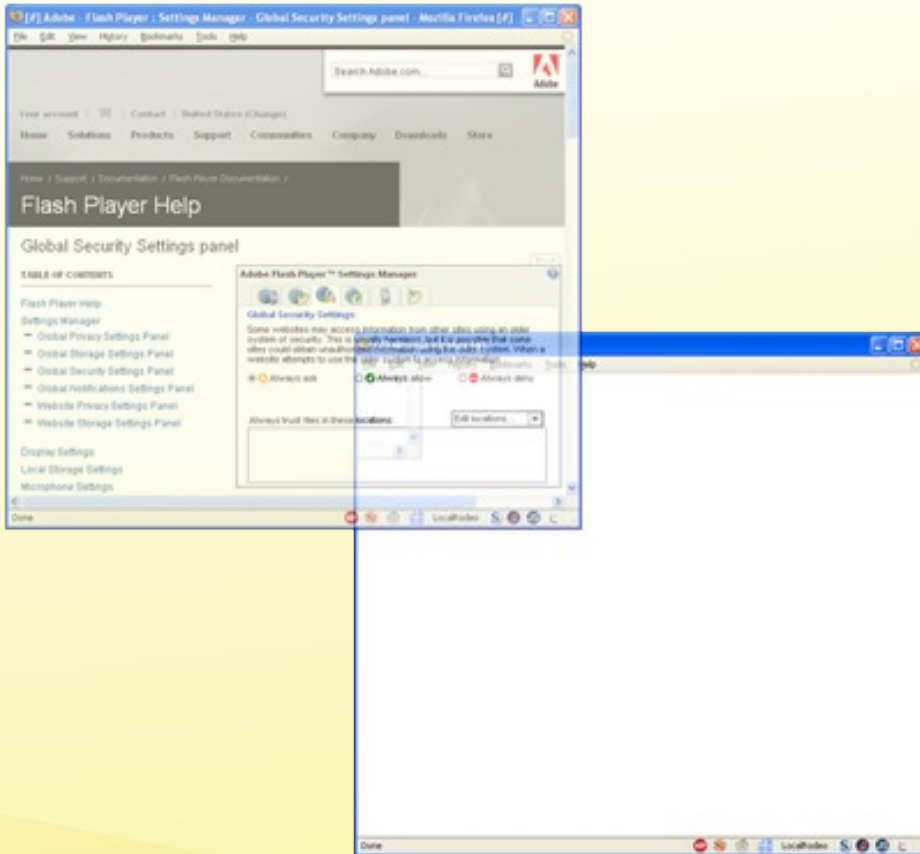
Clickjacking praktisch (4)

1. Komponente des Angriffs: Die Zielseite



Clickjacking praktisch (5)

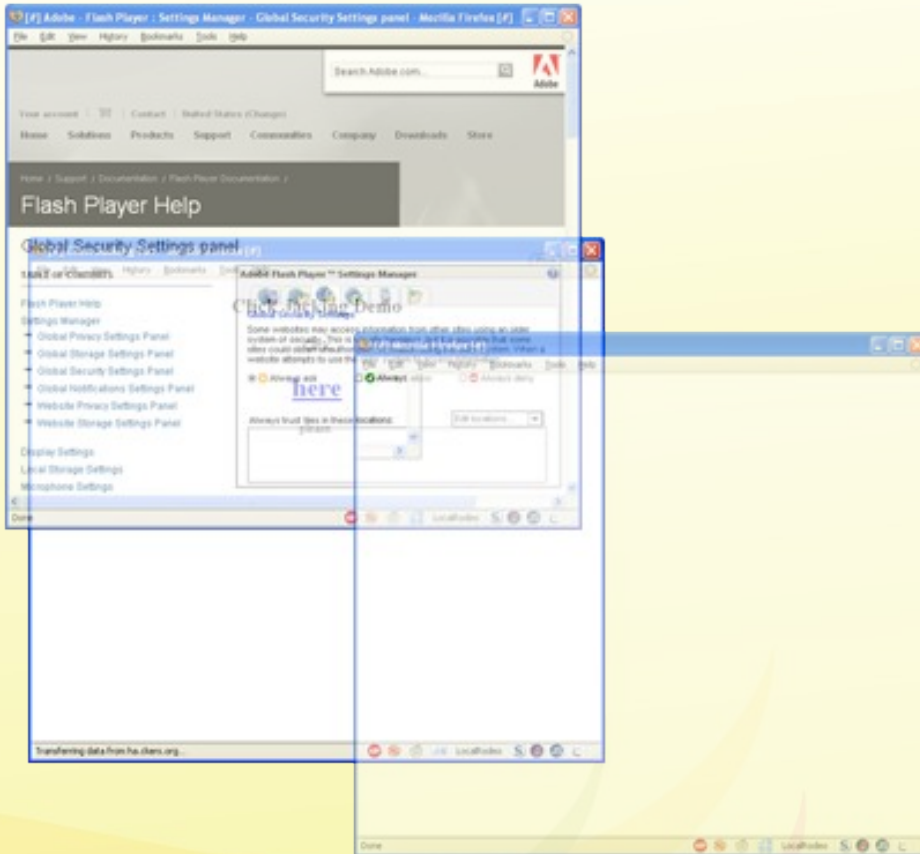
Zielseite so in iFrame platzieren, das der Ziel-Link oder Knopf in der oberen linken Ecke liegt



Clickjacking praktisch (6)

Diese Seite als iFrame in die Seite einfügen, die der Benutzer sieht

iFrame enthält nur Ziel-Link oder -Knopf, alles andere ist außerhalb

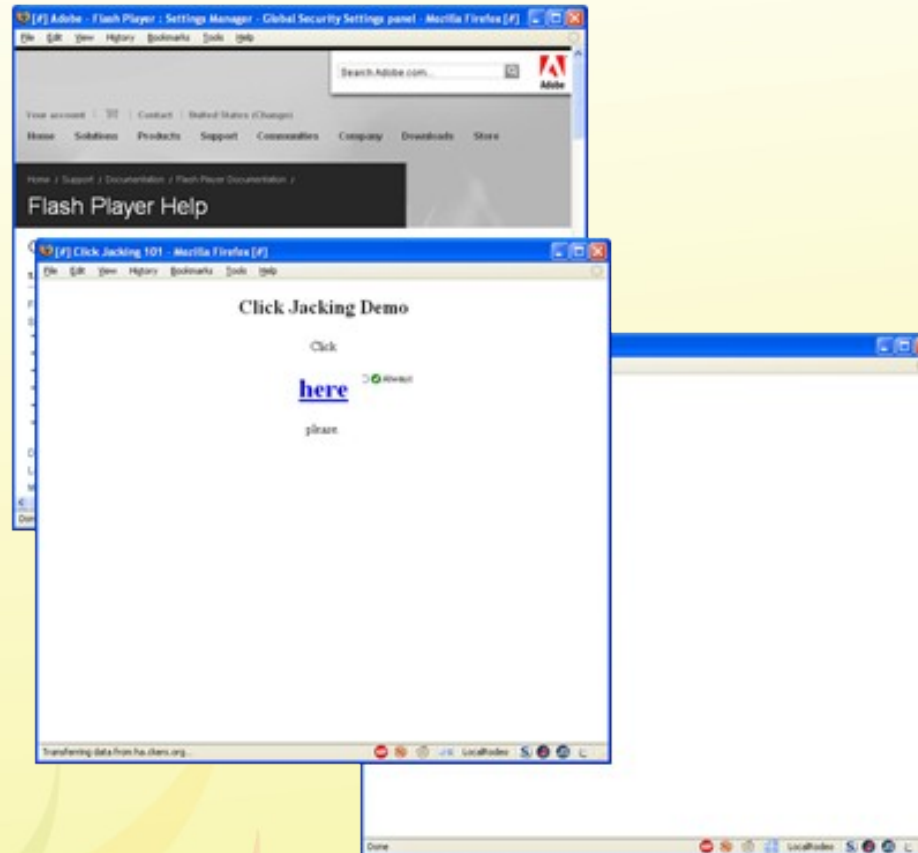


Clickjacking praktisch (7)

Danach iFrame so präparieren, das er unter dem Mauszeiger schwebt - egal wohin geklickt wird, die Maus ist an der richtigen Stelle

Zuletzt Opacity des den iFrame einschließenden div-Tags so ändern, das er unsichtbar wird

Clickjacking praktisch (8)



Clickjacking Vorschau (1)

Weitere mögliche Angriffe:

- Klick-Betrug (Click Fraud)
benötigt viele verschiedene IP-Adresse mit unterschiedlichen Browsern - Clickjacking liefert genau das
- Manipulation von ActiveX-Controls, Java-Applets, Ajax-Anwendungen, ...

Clickjacking Vorschau (2)

Bisher:

Benutzer führt Aktion aus

Aber ist das alles?

Virtuelle Tastatur, darüber ein unsichtbares
Doppel davon

Benutzer tippt PIN ein - auf der Tastatur des
Angreifers

Clickjacking Vorschau (3)

Nachteil:

Da die Klicks auf der Seite des Angreifers landen, passiert nichts auf der richtigen Seite

Aber fällt das wirklich auf?

Ist das nicht nur eine weitere Macke des Rechners/Programms/...?

Tastatur wird nach Erfolg freigegeben - beim zweiten Versuch klappt alles

Clickjacking verhindern (1)

Websites: Framebuster verwenden

```
<script type="text/javascript">  
if (top!=self)  
    top.location.href=self.location.href;  
</script>
```

Nachteile;

- Funktionieren nicht ohne JavaScript, Clickjacking schon
- Funktioniert nicht immer

Clickjacking verhindern (2)

Benutzer:

- Firefox: NoScript enthält die Funktion ClearClick
(macht verborgene, durchsichtige und sonstwie getarnte Dialoge beim Anklicken sichtbar und fragt nach)
- Andere Browser: Nichts zu machen

Agenda

- Vorbemerkungen
- Klassiker: XSS und CSRF
- SQL-Injection
- Presentation Layer
- Clickjacking
- **„Dies und Das“**

Client-Quelltext (1)

Angreifer hat Zugriff auf den Client-Quelltext

- Obfuscating ist keine Verschlüsselung
- Verschlüsselung ist nicht möglich

„Alles, was Sie in den Client schreiben, kann und wird gegen Sie verwendet“

Client-Quelltext (2)

- Kommentare liefern wertvolle Informationen für weitere Angriffe
- „Versteckte“ Informationen ebenso
- Und Anwendungslogik wie z.B. Rabattgruppen für Warenkörbe lassen sich gleich direkt anwenden

Race Condition (1)

Ajax ist Asynchron:

Mindestens 2 Threads auf dem Server:

Einer verarbeitet die Daten,

Einer kommuniziert mit dem Client

Das kann zu Race Condition führen

Race Condition (2)

Race Condition klassisch:

Anwendung erzeugt temporäre Datei

Angreifer ersetzt die durch symbolischen Link

Anwendung manipuliert dessen Ziel

Race Condition (3)

Race Condition Allgemein:

Anwendung ist implizit auf bestimmte Reihenfolge von Aktionen angewiesen, ohne das explizit zu erzwingen

Race Condition (4)

Race Condition im Web 2.0

Beispiel Webshop

Auf dem Server zwei Funktionen:

- `inDenWarenkorbLegen ()`
`WareHinzufügen ()`
`SummeBerechnen ()`
- `zurKasseGehen ()`
`KarteBelasten ()`
`WareLiefern ()`

Race Condition (5)

Für 2 Artikel:

Client:

- `inDenWarenkorbLegen ()`
- `inDenWarenkorbLegen ()`
- `zurKasseGehen ()`

Race Condition (6)

Für 2 Artikel:

Server:

WareHinzufügen ()

SummeBerechnen ()

WareHinzufügen ()

SummeBerechnen ()

KarteBelasten ()

WareLiefern ()

Race Condition (7)

Anwendungslogik im Client sorgt dafür, das immer erst mindestens einmal `inDenWarenkorbLegen()` aufgerufen wurde, bevor `zurKasseGehen()` aufgerufen wird

Ein Angreifer kann diese Prüfung umgehen

Erwischt er den richtigen Zeitpunkt, kann er die Reihenfolge der Unterfunktionen auf dem Server zu seinen Gunsten ändern

Race Condition (8)

WareHinzufügen ()

SummeBerechnen ()

WareHinzufügen () <===

KarteBelasten ()

SummeBerechnen () <===

WareLiefern ()

Den 2. Artikel gab es kostenlos

Race Condition verhindern

Gegenmaßnahme:

Locks schließen zusammengehörende Aktionen ein

Gefahr: Deadlock

Race Condition verhindern

```
SetzeLock ()  
  WareHinzufügen ()  
  SummeBerechnen ()
```

```
LöseLock ()
```

```
SetzeLock ()  
  WareHinzufügen ()  
  SummeBerechnen ()
```

```
LöseLock ()
```

```
SetzeLock ()  
  KarteBelasten ()  
  WareLiefern ()
```

```
LöseLock ()
```

Fragen?

Vielen Dank...

... für Ihre Aufmerksamkeit

Material und Links auf

www.ceilers-it.de/konferenzen/

AJAX IN ACTION

Konferenz für Web Development, Design & Technology