Security

Web App und Ruby on Rails Security

Veranstaltung: Web-Engineering 1
FH-Münster - Bachelor Wirtschaftsinformatik
Semester: 4

Dave Kaufmann	737472
Nico Lindmeyer	737045
Daniel Reider	734544
Johann Schäfer	736694

Präsentation am 11.06.2014

Agenda

- Einleitung
- Angriffsszenarien
- Generelle Schutzmöglichkeiten
- Fazit

Agenda

- Einleitung
  - Motivation
  - Strafrechtliche Aspekte
- Angriffsszenarien
- Generelle Schutzmöglichkeiten
- Fazit

Motivation

- Warum ist Web-Application-Security heutzutage immer wichtiger?
  - Immer mehr Daten liegen im Internet
  - Viele persönliche und sensible Daten auf verschiedenen Webseiten hinterlegt
  - Cloud computing

Strafrechtliche Aspekte

- Vorbereiten des Ausspähens und Abfangens von Daten
  - Bis zu einem Jahr Haft / Geldstrafe
- Datenveränderung / Computersabotage
  - Bis zu 5 Jahren Haft / Geldstrafe
- Ausspähen von Daten
  - Bis zu 3 Jahren Haft / Geldstrafe
- Computerbetrug
  - Bis zu 5 Jahren Haft / Geldstrafe

Agenda

- Einleitung
- Angriffsszenarien
  - SQL Injection
  - XSS
  - DoS / DDoS
  - CSRF
- Generelle Schutzmöglichkeiten
- Fazit

SQL-Injection

SQL-Injection

- Beschreibt das Verändern/Auslesen einer DB über eine Weboberfläche/die URL oder das Ausführen von eigenem Code
- Einfache SQL Abfrage
- Wird anstatt des Übergabeparameters übergeben

SQL-Injection

- Über die Eingabe:

login.sql

```
1   SELECT * FROM users
2   WHERE username = ' 'or 1=1 -- AND password = ""
```

SQL-Injection

- Über die URL

hp?category=2UNION ALL 1' AND 1=(SELECT COUNT(*) FROM tablenames); --

- Live-Hack SQLi

SQL-Injection - Prävention

- Datenbank-User nur mit erforderlichen Rechten
- Nur sinnvolle Werte zulassen: `If(is_numeric ($input )){…}`
- PHP: `$_REQUEST["$key"] = mysql_real_escape_string($val);`

SQL-Injection – Ruby on Rails

- Automatische Maskierung des Standards durch ActiveRecord

- Bei eigenen Bedingungen oder SQL-Abfragen muss die Maskierung manuell erfolgen

flight.rb

```
1  Flight.find(:all, :conditions => "code = '#{@params['code']}'")
2
```

SQL-Injection – RoR Maskierung

- Unsicher:

product.rb

```ruby
@products = Product.order("name #{direction}")
```

- Sicher:

product.rb

```ruby
direction = params[:direction] == "desc" ? "desc " : "asc"
@products = Product.order("name #{:direction]}")
```

Cross Site Scripting

XSS

Was ist XSS?

- Webseite nimmt Daten vom Nutzer an
- Sendet diese Daten weiter, **ohne** Prüfung

- ➔Skripte/Schadcode wird indirekt an das Opfer gesendet

Beispiel persistentes XSS

Mein Gästebuch

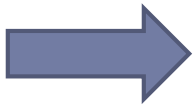
Name:

h4x0r

Text:

```
<script>alert("Hello World!");</script>
```

Write



Mein Gästebuch

New Entry

h4x0r:

Hello World!

OK

Nicht persistentes XSS

- Webseite nimmt Argument in der URL an
  - www.mysite.de/search.html?**arg=Katzenbilder**
- Ausgaben nach dem Muster:
  - Sie suchen nach: **Katzenbilder**
- Wir geben ein:
  - ?arg=**<script>alert(„XSS“);</script>**
- Dadurch würde sich ergeben
  - Sie suchen nach: **<script>alert(„XSS“);</script>**

- **Aber:** Javascript wird auf dem Client ausgeführt

DOM-basiert / lokales XSS

- Server ist nicht beteiligt
- Parameter wird mit clientseitiger Skriptsprache ausgelesen

Gefahren von XSS

- Cookie Stealing
- Redirect
- Design Änderungen
- Etc.

Schutz vor XSS

- Alle Eingaben des Nutzers als unsicher betrachten
  - Never trust parameters from the scary internet
- Whitelist anstatt Blacklist

- Metazeichen ersetzen
  - RoR Beispiel: <%=h @post.message %>

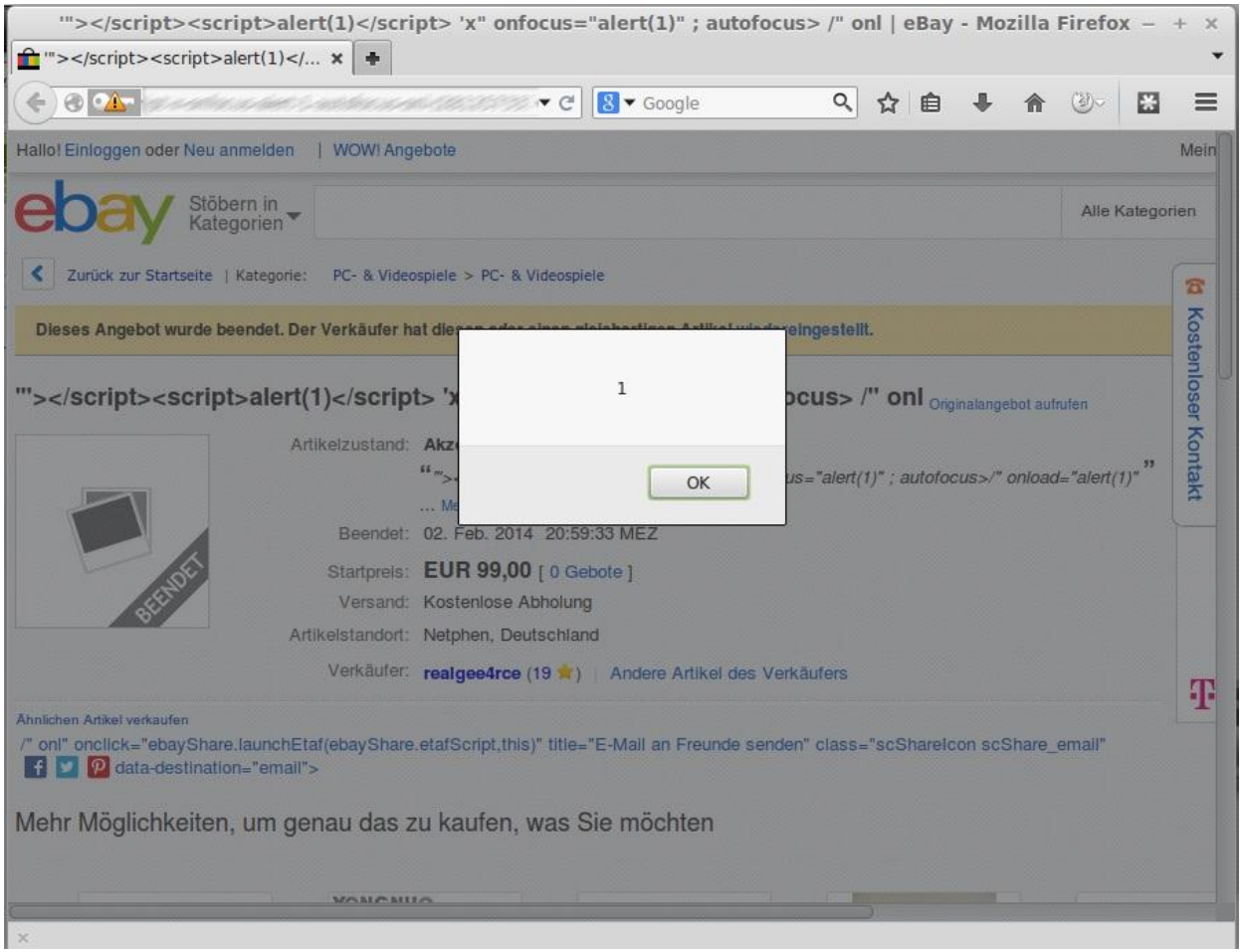
- Lokales XSS müsste Clientseitige Prüfung stattfinden
  - → Wieder leicht manipulierbar

Live-Hacking XSS

- XSS vulnerable guestbook

Aktuelle XSS Vorfälle

Ebay – 23.05.2014

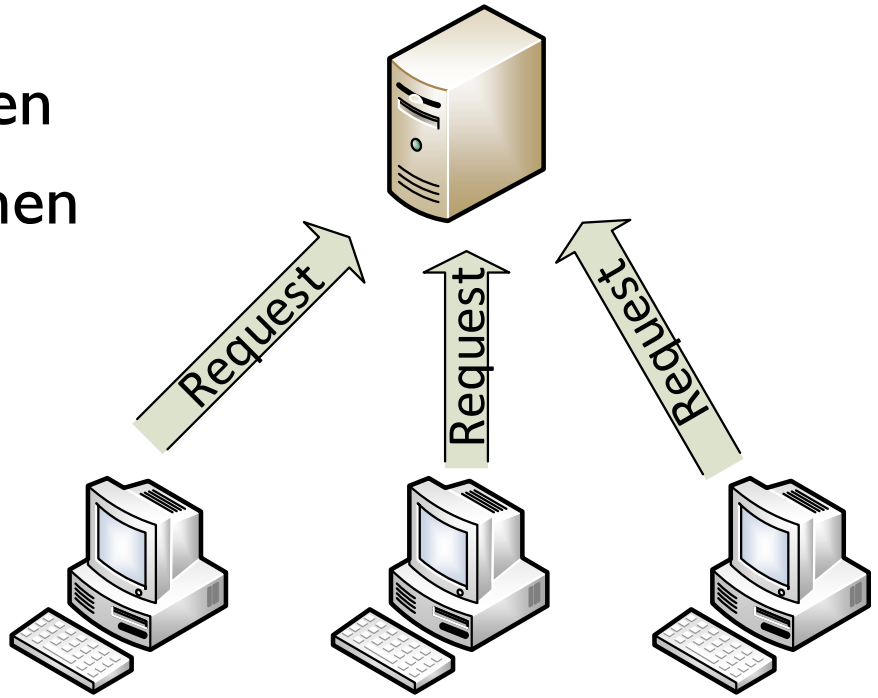


(Distributed) Denial of Service Attacken

DoS / DDoS

Was ist eine DoS / DDoS Attacke

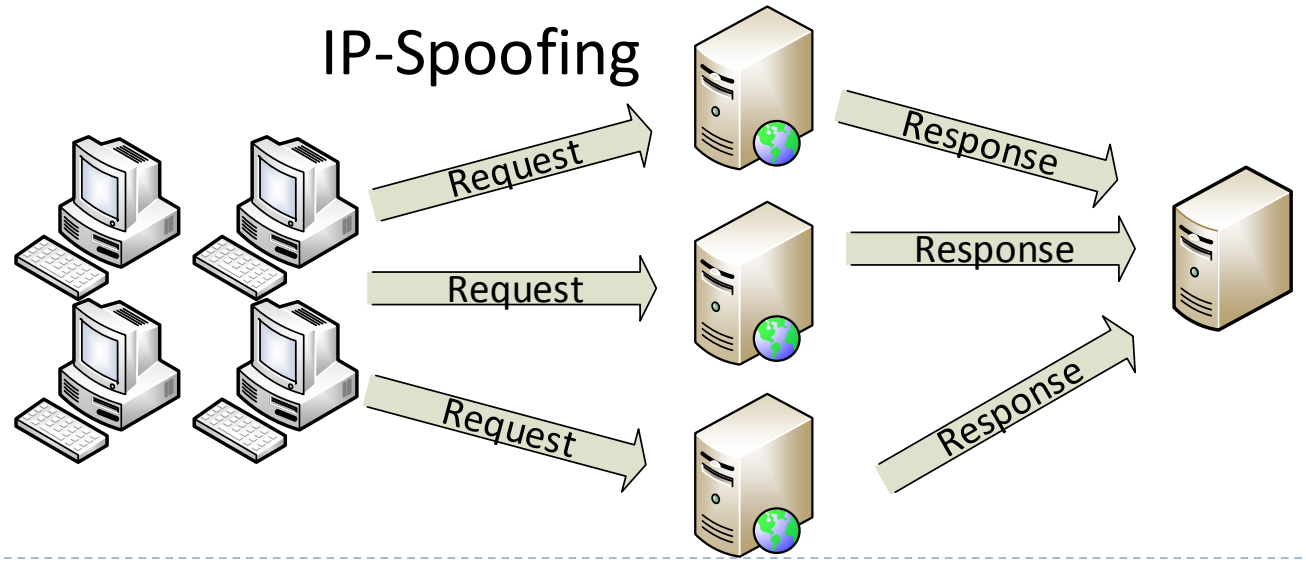
- Überlastung der Infrastruktur
  - Internetzugang
  - Betriebssystem
  - Dienst (z.B. HTTP)
- Größere Anzahl von Anfragen als verarbeitet werden können



Distributed-Reflected-Denial-of-Service-Angriff (DRDoS)

- Indirekter Angriff auf das Opfer
- Anfragen an normal arbeitende Internet Dienste
- Ändern der Absender IP (IP-Spoofing)

- Angreifer kann nicht direkt ermittelt werden



Schutz vor DoS / DDoS

- Sperrlisten von IP-Adresse
- Analysen und Filterung auf Routern
- Ändern der IP-Adresse
- Load Balancing (Server cluster)

Aktuelle DoS Anriffe

- Webseite der Stadt Frankfurt 18. Mai 2012
  - Durch Anonymous im Rahmen der Blockupy-Proteste
- Feedly (RSS-Dienst) 11.06.2014
  - DDoS Attacke
  - Erpressung des Angreifers

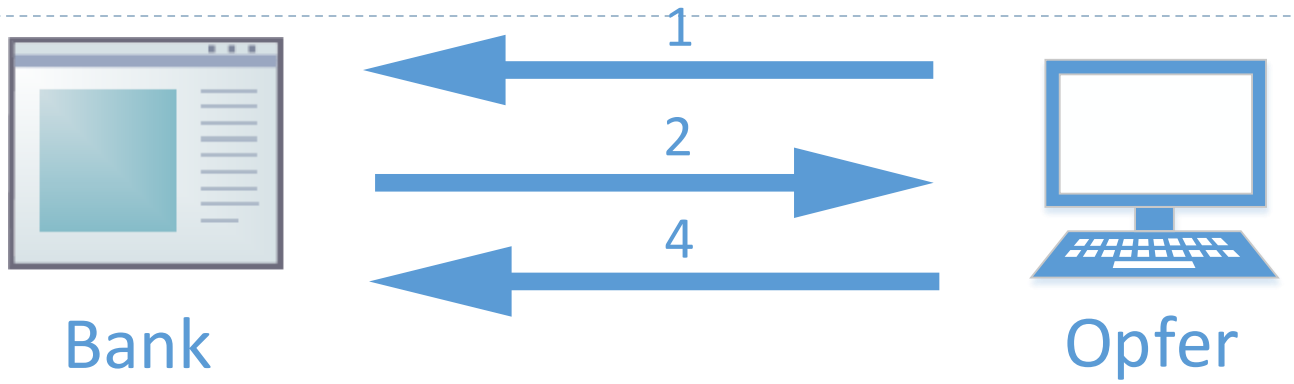
Cross Site Request Forgery

CSRF

CSRF

- Cross Site Request Forgery
  - XSRF
  - Session Riding
- User muss legitime Seite aufsuchen und sich einloggen
- Angreifer benutzt gültige Session des Opfers
- Durch Image-Tags, XSS oder andere Techniken löst das Opfer unbeabsichtigt einen gefälschten HTTP-Request für eine Anwendung aus

CSRF



1. Opfer loggt sich auf der Seite seiner Bank ein
2. Cookie wird generiert
3. Opfer besucht eine Seite mit einem verstecktem Image Element

<img src="http://www.bank.de/
ueberweisung.php&betrag=1000&konto=123456" />

4. Bank erkennt die Session als gültig und führt die Transaktion durch

Schutzmöglichkeiten

- Web Application Firewalls

- CRSF-Token
  - Zusätzliches verstecktes Feld mit enthaltener Zufallszahl
  - Jeder Benutzer ein anderes Token
  - Wird jedesmal neu generiert, wenn sich der Benutzer einloggt
  - Skripte überprüfen den empfangenen Token und vergleichen diesen mit dem bereits in der Session gespeicherten
  - Bei Abweichung, verweigert das Skript die Ausführung

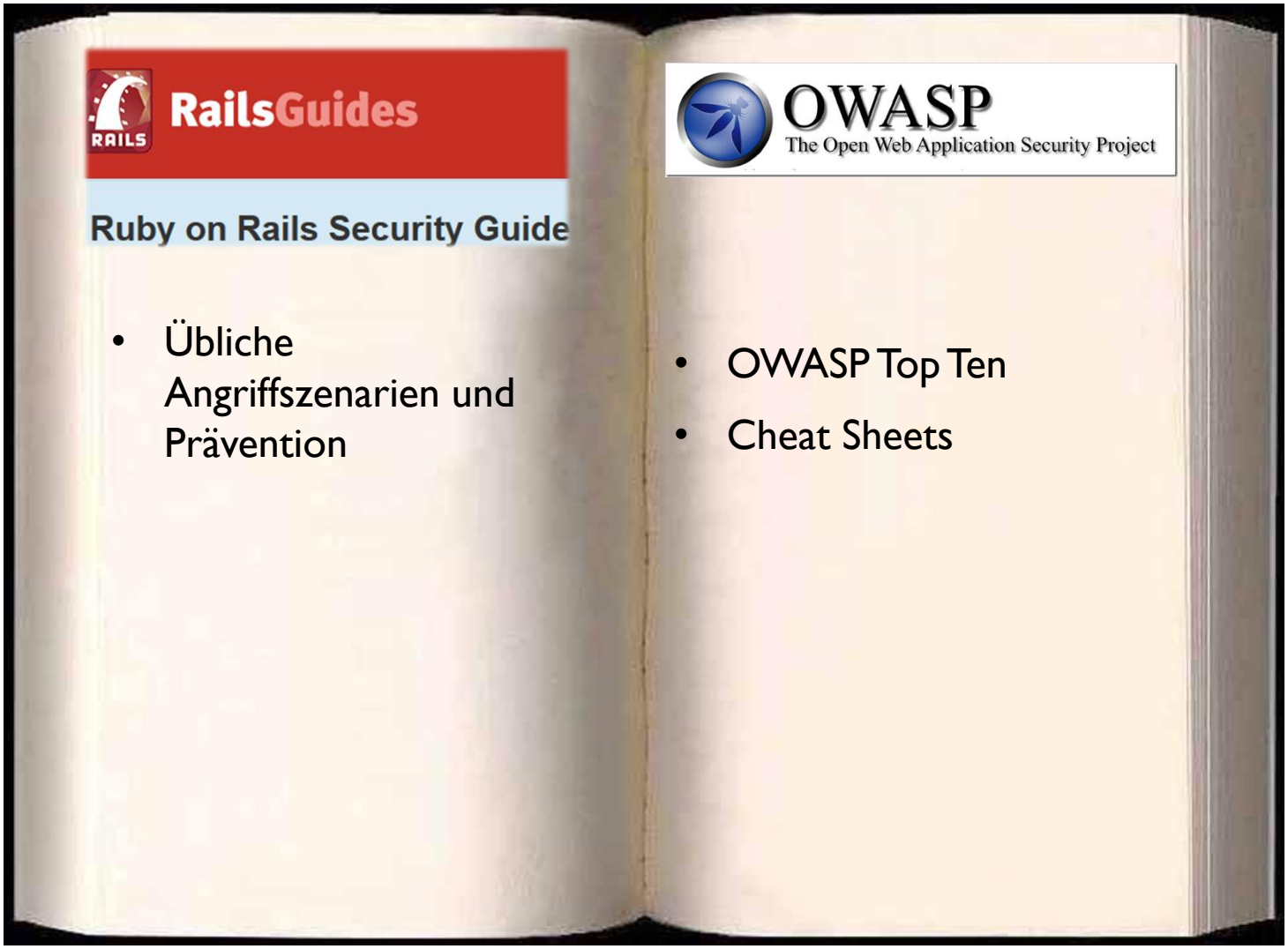
Agenda

- Einleitung
- Angriffsszenarien
- Generelle Schutzmöglichkeiten
  - Allgemeines
  - Brakeman
- Fazit

Generelle Schutzmöglichkeiten

Wissen!

Generelle Schutzmöglichkeiten



- Übliche Angriffszenarien und Prävention

- OWASP Top Ten
- Cheat Sheets

Generelle Schutzmöglichkeiten

Wissen!

Up to date bleiben!

Generelle Schutzmöglichkeiten

Security News!

- Security Mailing lists
- www.cvedetails.com
- Updates

Rubyonrails » Ruby On Rails : Security Vulnerabilities

CVSS Scores Greater Than: 0 1 2 3 4 5 6 7 8 9
Sort Results By : CVE Number Descending CVE Number Ascending CVSS Score Descending Number Of Exploits Descending
Copy Results Download Results Select Table

#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score	Gained Access Level	Access	Complexity
1	CVE-2014-0130	22		Dir. Trav.	2014-05-07	2014-05-31	4.3	None	Remote	Medium
Directory traversal vulnerability in actionpack/lib/abstract_controller/base.rb in the implicit-render implementation in Ruby on Rails before 3.2.18, route globbing configurations are enabled, allows remote attackers to read arbitrary files via a crafted request.										
2	CVE-2014-0082	20		DoS	2014-02-20	2014-04-24	5.0	None	Remote	Low
actionpack/lib/action_view/template/text.rb in Action View in Ruby on Rails 3.x before 3.2.17 converts MIME type strings to symbols during use of remote attackers to cause a denial of service (memory consumption) by including these strings in headers.										
3	CVE-2014-0081	79		XSS	2014-02-20	2014-03-26	4.3	None	Remote	Medium
Multiple cross-site scripting (XSS) vulnerabilities in actionview/lib/action_view/helpers/number_helper.rb in Ruby on Rails before 3.2.17, 4.0.x befo attackers to inject arbitrary web script or HTML via the (1) format, (2) negative_format, or (3) units parameter to the (a) number_to_currency, (b) helper.										
4	CVE-2014-0080	89		Exec Code Sql	2014-02-20	2014-02-20	6.8	None	Remote	Medium
SQL injection vulnerability in activerecord/lib/active_record/connection_adapters/postgresql/cast.rb in Active Record in Ruby on Rails 4.0.x before allows remote attackers to execute "add data" SQL commands via vectors involving \ (backslash) characters that are not properly handled in oper										

Generelle Schutzmöglichkeiten

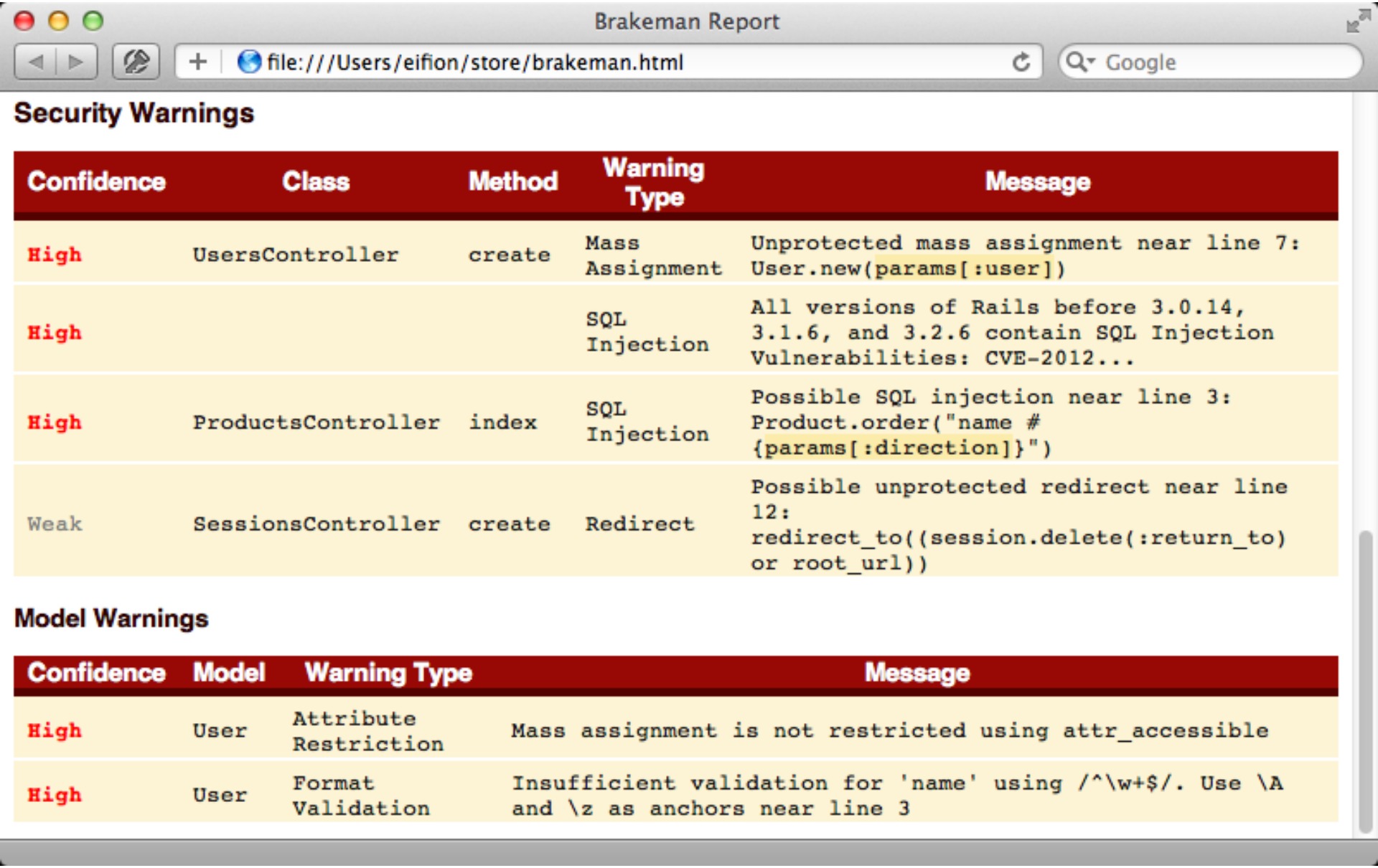
Wissen!

Tools!

Up to date bleiben!

Brakeman

- Open Source Gem für RoR Anwendungen
- Sucht nach Sicherheitsschwachstellen
  - XSS, SQL-Injection, CSRF, Mass Assignment etc.
- Aktuelle Version: 2.6.0 (Release: 06.06.2014)
- Beispiel:
  - `gem install brakeman`
  - `brakeman`
  - `brakeman -o brakeman.html`
  - `www.rails-brakeman.com`

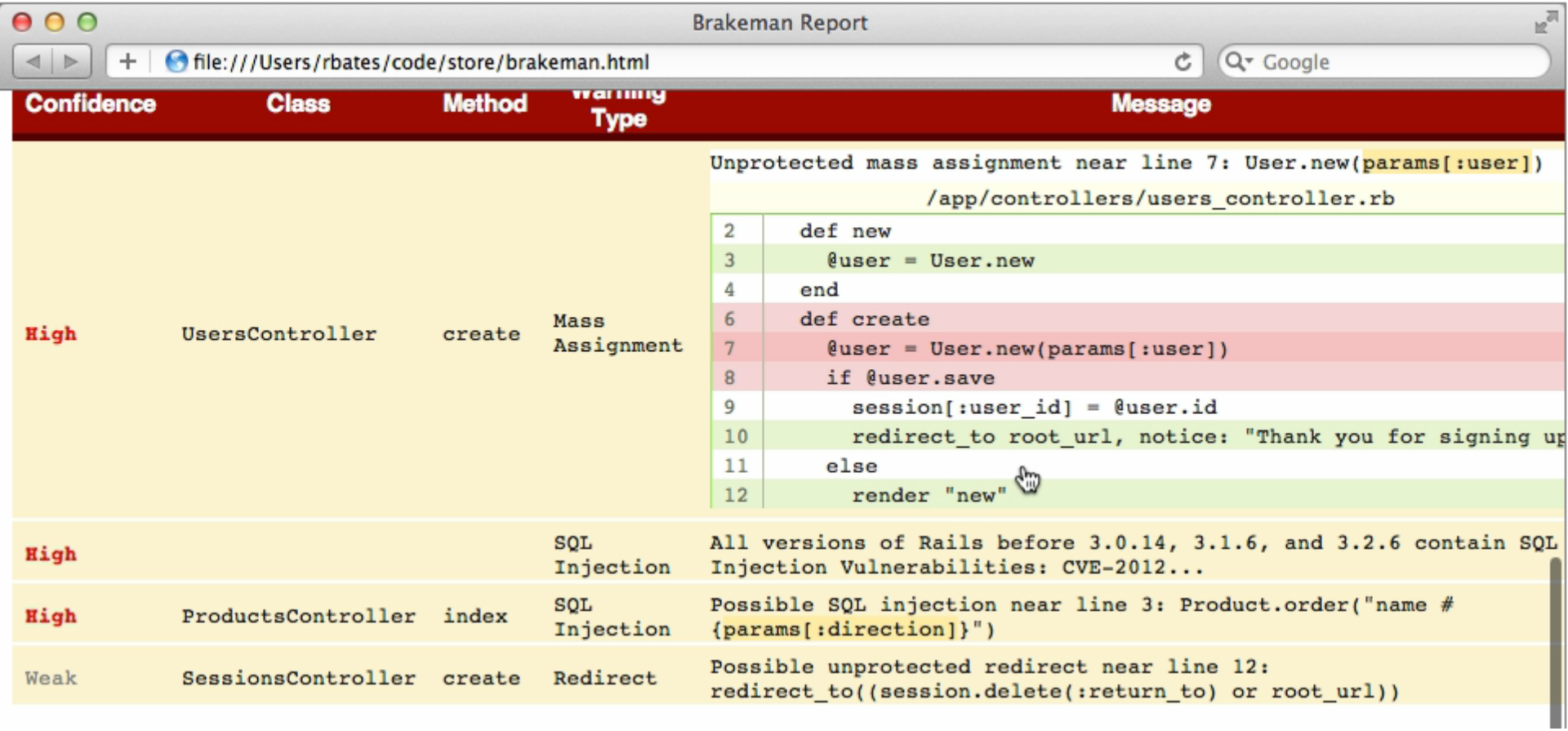


Security Warnings

Confidence	Class	Method	Warning Type	Message
High	UsersController	create	Mass Assignment	Unprotected mass assignment near line 7: User.new(params[:user])
High			SQL Injection	All versions of Rails before 3.0.14, 3.1.6, and 3.2.6 contain SQL Injection Vulnerabilities: CVE-2012...
High	ProductsController	index	SQL Injection	Possible SQL injection near line 3: Product.order("name #{params[:direction]}")
Weak	SessionsController	create	Redirect	Possible unprotected redirect near line 12: redirect_to((session.delete(:return_to) or root_url))

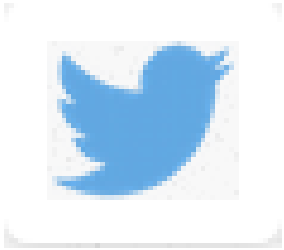
Model Warnings

Confidence	Model	Warning Type	Message
High	User	Attribute Restriction	Mass assignment is not restricted using attr_accessible
High	User	Format Validation	Insufficient validation for 'name' using /^\w+$/. Use \A and \z as anchors near line 3


Brakeman Report
file:///Users/rbates/code/store/brakeman.html
Google
Confidence
Class
Method
Warning Type
Message
High
UsersController
create
Mass Assignment
Unprotected mass assignment near line 7: User.new(params[:user])
/app/controllers/users_controller.rb
2 def new
3 @user = User.new
4 end
6 def create
7 @user = User.new(params[:user])
8 if @user.save
9 session[:user_id] = @user.id
10 redirect_to root_url, notice: "Thank you for signing up
11 else
12 render "new"
High
SQL Injection
All versions of Rails before 3.0.14, 3.1.6, and 3.2.6 contain SQL Injection Vulnerabilities: CVE-2012...
High
ProductsController
index
SQL Injection
Possible SQL injection near line 3: Product.order("name #{params[:direction]}")
Weak
SessionsController
create
Redirect
Possible unprotected redirect near line 12: redirect_to((session.delete(:return_to) or root_url))

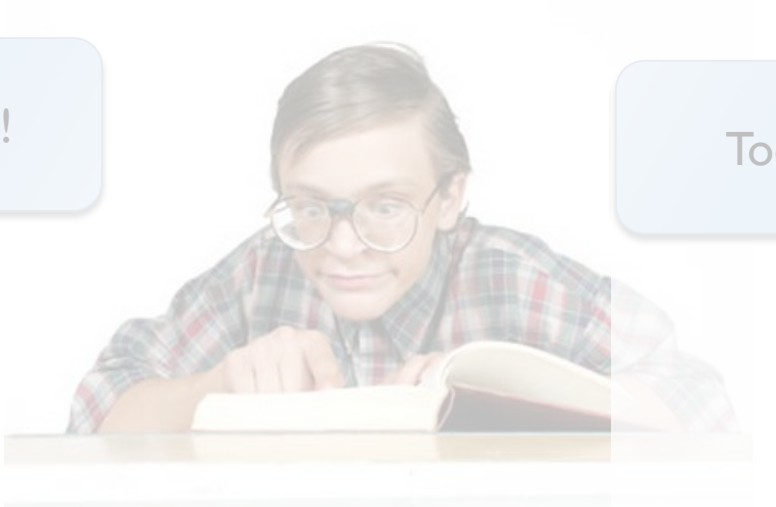
Brakeman

- Änderungsbeispiele in der Dokumentation auf *brakemanscanner.org*
- Keine 100%ige Sicherheitsgarantie
- Nutzer von Brakeman



Generelle Schutzmöglichkeiten

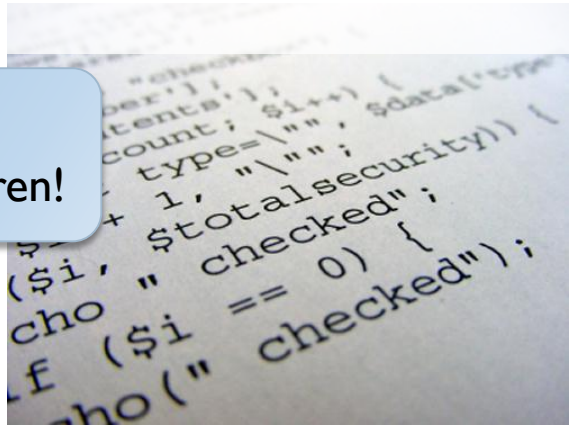
Wissen!



Tools!

Up to date bleiben!

Sicheres Programmieren!

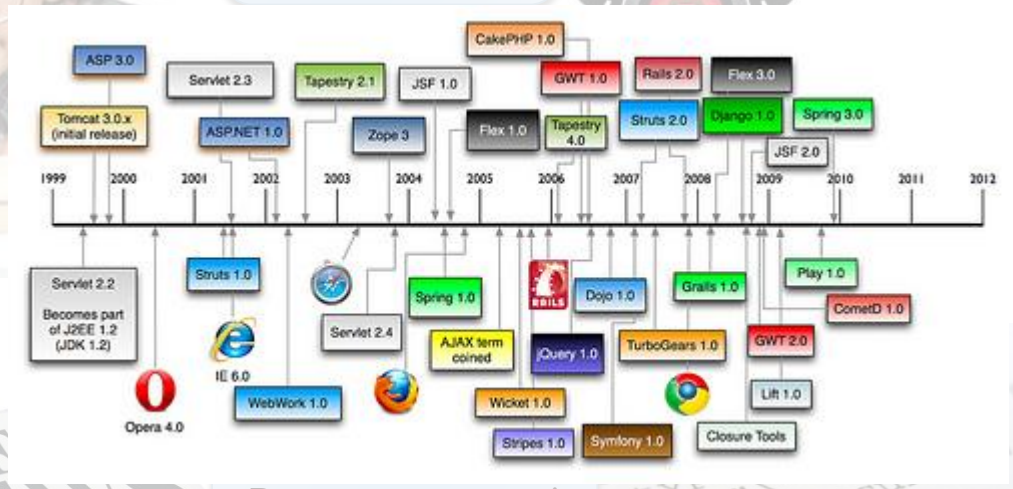


Generelle Schutzmöglichkeiten

Wissen!

Tools!

Frameworks!



Up to date bleiben!

Programmieren!

Frameworks

ASP.NET

CanCan

devise

- Authentifizierung
- Autorisierung
- Session-Management
- Kryptographie

Agenda

- Einleitung
- Angriffsszenarien
- Generelle Schutzmöglichkeiten
- Fazit

Fazit

- Vielzählige Angriffsmöglichkeiten
- 100 %ige Sicherheit nicht möglich
- Ständiger Aktualisierungsbedarf
- Gute Unterstützung durch Frameworks / Tools
- Hohes Gefahrenpotential bei Fahrlässigkeit

Vielen Dank für Ihre Aufmerksamkeit

Fragen?