

Ruby on Rails Sicherheit

Heiko Webers
42@rorsecurity.info

Heiko Webers

- Ruby On Rails Security Project: www.RoRSecurity.info
- E-Book „Ruby On Rails Security“
- Ruby On Rails Security Audits
- Webanwendungen



Trends



- Motivation der Cracker: Geld verdienen
- Neuerdings: Angriffe auf beliebte Portale
 - Einbruch in den Server: Apache, cPanel, CMS, SSH
 - Bannerwerbung mit Malware (Yahoo's Right Media Ad Network)

Trends

- Angriffe auf beliebte Portale
 - Exploits für bestimmte Sicherheitslücken (MySpace: Trojaner in QuickTime)
 - Ganze Frameworks: Mpack, 500 – 1000\$, aus dem russischen Untergrund, 40-50% garantierte Erfolgsquote
- Hier: Sicherheit der (oberen) Anwendungsschicht

Injection

- Falsch
 - Benutzereingaben direkt verwenden
- Richtig
 - Benutzereingaben sind schädlich - bis das Gegenteil bewiesen ist
 - Filtern, oder besser escapen
 - Passend zum Kontext

Kontext

- SQL – SQL-Injection in `find_by_sql()`
- SGML (HTML, XML, RSS...) – Cross-Site Scripting (JavaScript z.B. im Benutzernamen), SafeErb plug-in
- JavaScript – schädlicher Code in RJS
 - `escape_javascript()`

Kontext

- CSS – Samy-Wurm bezwingt MySpace:
- `<div id=mycode style="BACKGROUND: url(' javascript:eval(document.all.mycode.expr) ') " expr="..." />`

Kontext

- Kommandozeilen-Parameter: Kein | und ; erlaubt
- `system()` anstatt ``command``
 - ``ls #{dir}` # dir #="whatever | rm *"`

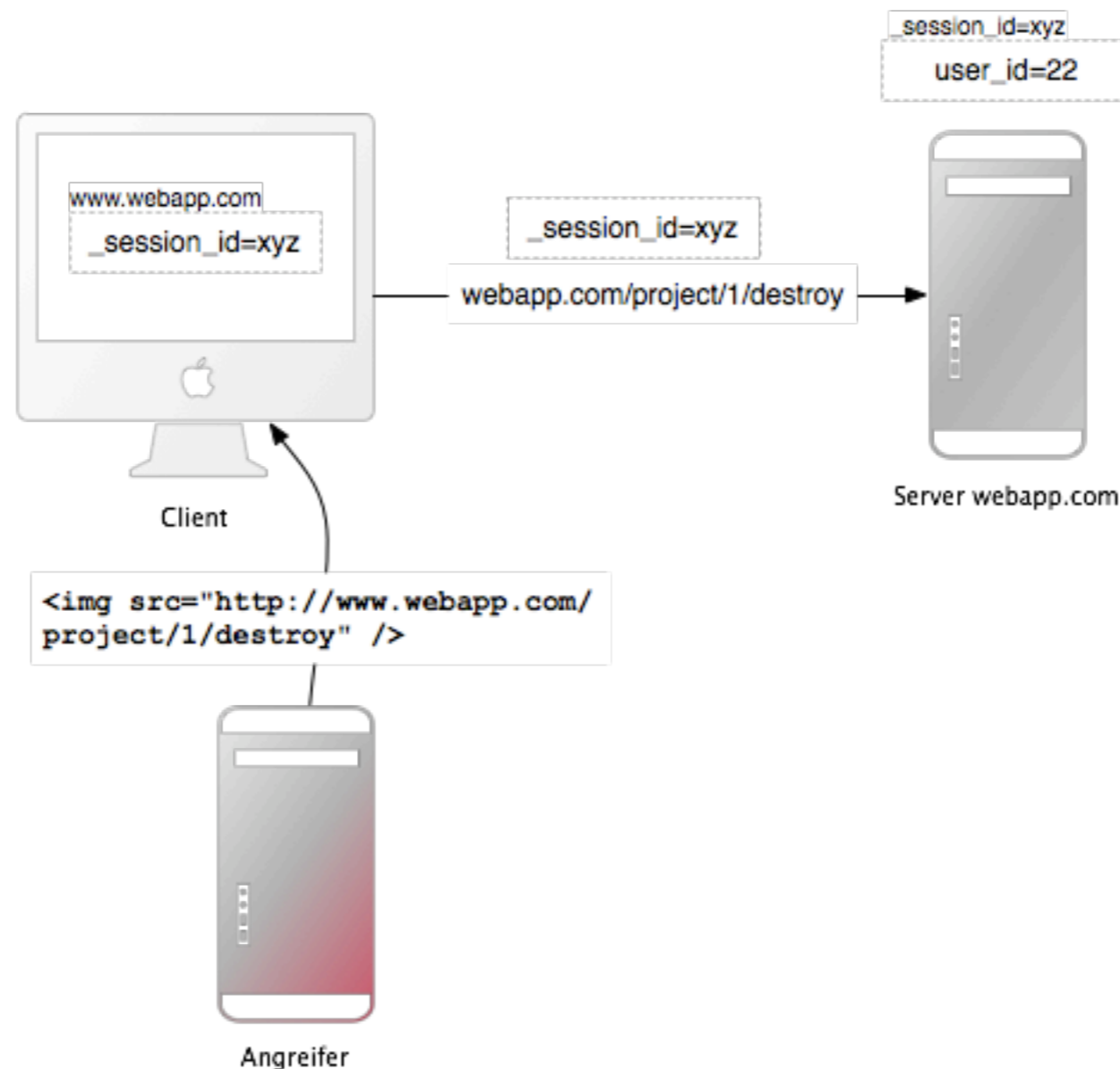
Kontext

- Textile – Ergebnis mit `sanitize()` filtern
 - `!bunny.gif(Bunny" onclick="alert(1))!`
 - `<p></p>`

Blacklist - Whitelist

- `before_filter :only` anstatt `:except`
- `Attr_accessible` anstatt `attr_protected`
- XSS: `` erlauben, nicht `<script>` entfernen
- Nicht Eingaben mit Blacklists „korrigieren“:
`"<sc<script>ript>" .gsub("<script>", "")`
- Eingabe zurückweisen

Cross-Site Reference Forgery (CSRF)



CSRF in Rails

- Falsch
 - `GET /project/1/destroy`
- Richtig
 - `GET /project/1/show`
 - `POST /project/1/destroy`
 - `verify :method => :post, :only => [:destroy]`

CSRF & POST

- `To the survey`
- ``

CSRF in Rails

- Richtig
 - `protect_from_forgery :secret => "min. 30 Zeichen"`
 - In `form_for()`, und anderen, wird automatisch eine Prüfsumme eingefügt
 - `<input name="authenticity_token" type="hidden" value="3a1e11299eff1fa5cbc724ca32978448098af0" />`

Admin-Bereich

- Falsch
 - XSS-Lücken (privilegiertes Cookie stehlen)
 - CSRF-Lücken
``
 - Das Cookie eines Admins aus der „normalen“ Anwendung gilt auch im Admin-Bereich

Admin-Bereich

- Richtig
 - Admin muss sich trotz gültigem Cookie einloggen
 - Eigener Account für Admin-Bereich
 - Sicherheit sollte hier noch ernster genommen werden (speziell XSS & CSRF)
 - Vorkehrungen für den schlimmsten Fall, Angreifer hat Zutritt zum Admin-Bereich

Admin-Bereich

- Richtig
 - admin.example.com anstatt www.example.com/admin
 - IP überprüfen: `request.remote_ip`

Login

- Falsch
 - Plugins nicht aktualisieren (z. B. `restful_authentication`)
- Richtig
 - Plugins und generierten Code aktualisieren

Login

- Sicherheitslücke in restful_authentication
 - `GET /activate?id=`
 - `User.find_by_activation_code(params[:id])`
 - `SELECT * FROM users WHERE
(users.`activation_code` IS NULL) LIMIT 1`
- Siehe http://www.rorsecurity.info/2007/10/28/restful_authentication-login-security/

Benutzerverwaltung

- Falsch
 - Passwort und E-Mail sehr leicht zu ändern
 - Kann per CSRF von extern geändert werden
 - Oder bei gestohlenem Cookie
- Richtig
 - Einen Account zu übernehmen erschweren
 - Passwort muss bei wichtigen Änderungen eingegeben werden

Benutzerverwaltung

- Falsch
 - Mit eindeutige Fehlermeldungen beim Login und „Passwort-vergessen“ lassen sich Benutzernamen herausfinden
 - Benutzer-Liste + Wörterbuch = Brute-Force
- Richtig
 - Nach einigen fehlgeschlagenen Login-Versuchen Account sperren oder CAPTCHA

CookieStore

- Der Client sieht, was in der Session steht
 - `<base64 enkodierte Session>-<Prüfsumme>`
- Falsch
 - Geheimnisse, Daten $>$ 4KB, ganze Objekte
 - Ein triviales „Secret“ verwenden
- `config.action_controller.session = {:secret => 'trivial' }`

CookieStore

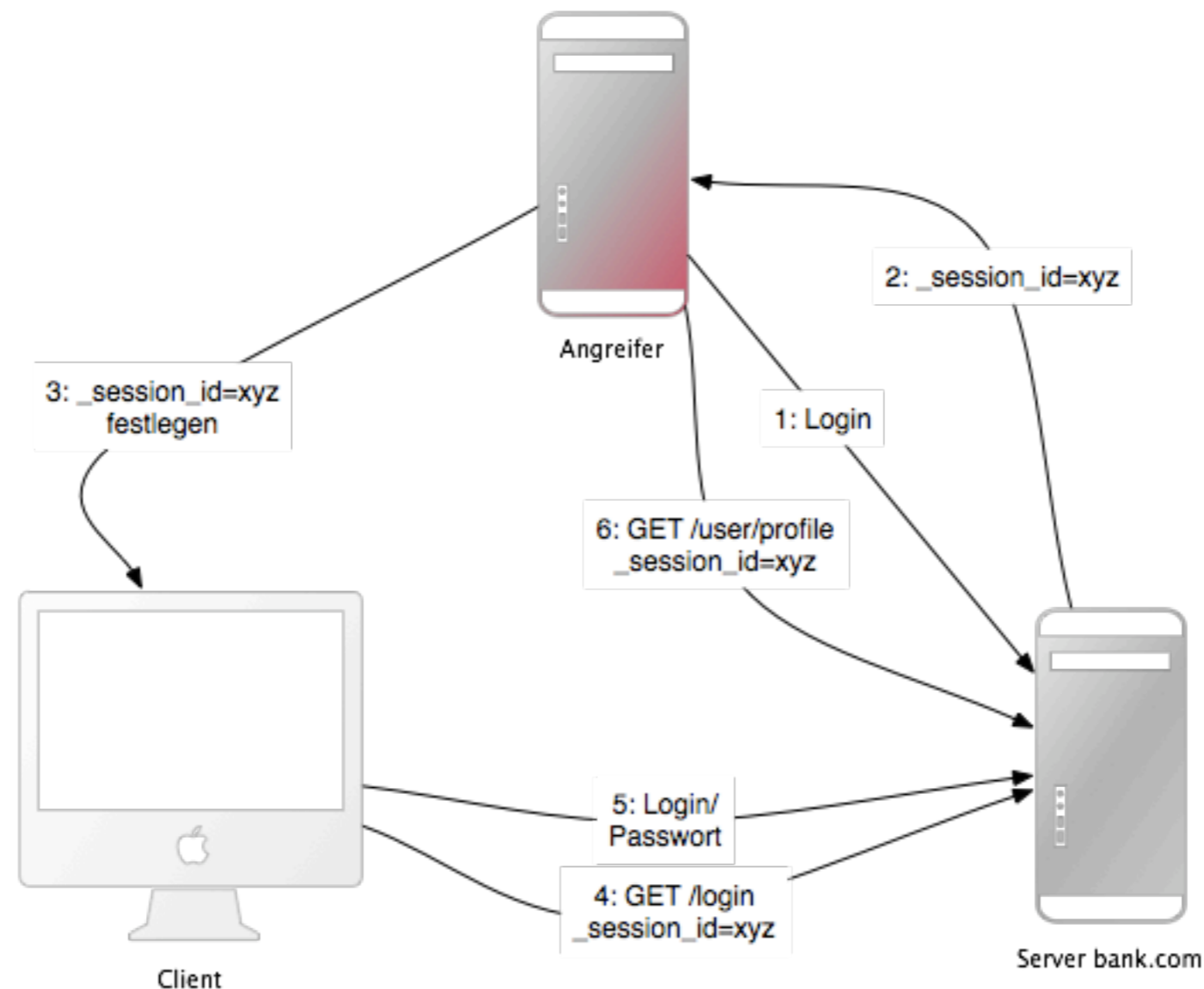
- Richtig
 - Ok, wenn nur `user_id` und `flash` in der Session
 - Mindestens 30 Zeichen für das „Secret“

CookieStore: Replay Attacken

- Benutzer hat Punkte, gespeichert in der Session
- Benutzer kauft etwas
- Die Punkte in der Session werden verringert
- Unehrlicher Benutzer hat Session von vor dem Kauf gespeichert und lädt sie in das Browser Cookie: Die Punkte sind zurück
- Normalerweise mit einem 1x gültigen Prüfwert in der Session gelöst. Schwer bei mehreren Mongrels

Session Fixation

- Anstatt ein Cookie zu stehlen, legt der Angreifer beim Opfer ein ihm bekanntes fest



Session Fixation

- Cookie festlegen:

```
document.cookie="_session_id=16d5b78abb28e3d62...";
```

- Falsch
 - XSS-Lücken: Einfachster Art Cookies festzulegen
 - Cookies, bzw. die enthaltene Session, nicht auslaufen lassen

Session Fixation

- Richtig
 - Bei geschlossenen Anwendungen: Den Angreifer nicht an eine gültige Session kommen lassen, z.B. auf der Login-Seite
 - Neue Session nach dem Login vergeben
`reset_session`

CAPTCHAs

- Hallo negative CAPTCHAs
 - Nicht der Benutzer muss beweisen, dass er ein Mensch ist, sondern ein automatischer Bot soll sich als solcher zu erkennen geben
 - Ein Honeypot-Feld per CSS verstecken, Bot füllt dieses aus
 - Wenn in diesem Feld ein Text steht, ist es ein Bot

Danke

Fragen?

Dateien

- Falsch
 - `send_file '/var/www/uploads/' + params[:filename]`
 - `GET /download?filename=../../../../etc/passwd`
 - `PUT /upload?filename=../../../../etc/passwd`
- Richtig
 - Dateinamen in der DB speichern, die Dateien nach der ID des Datensatzes benennen

Dateien

- Richtig
- Überprüfen, ob die Datei beim Download aus dem erwarteten Verzeichnis kommen
- `raise if DOWNLOAD_DIR != File.dirname(filename)`

Dateien

- Falsch
 - Uploads in Rails /public Verzeichnis speichern
 - Upload: /public/uploads/file.fcgi
- Richtig
 - Uploads nicht im Verzeichnis DocumentRoot von Apache speichern

Session Fixation

- Richtig
 - Sessions auslaufen lassen, Frequenz variabel
 - Aber: Ein Angreifer kann mit einem Skript die Session „am Leben halten“
 - Daher auch Sessions auslaufen lassen, die vor sehr langer Zeit erstellt wurden