1.– 4. September 2014
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Total abgemeldet – oder doch nicht?

Web Security rund um die Anmeldung

Dr. Stefan Schlott

BeOne Stuttgart GmbH

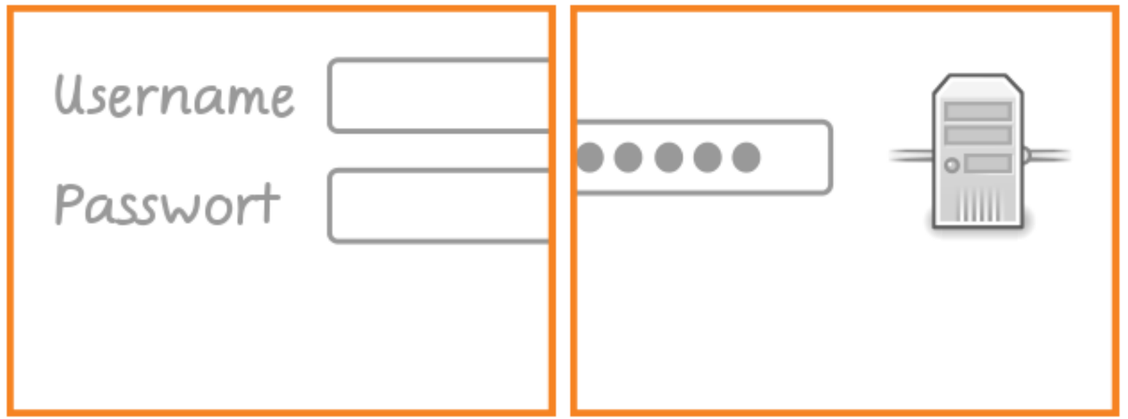
about.txt

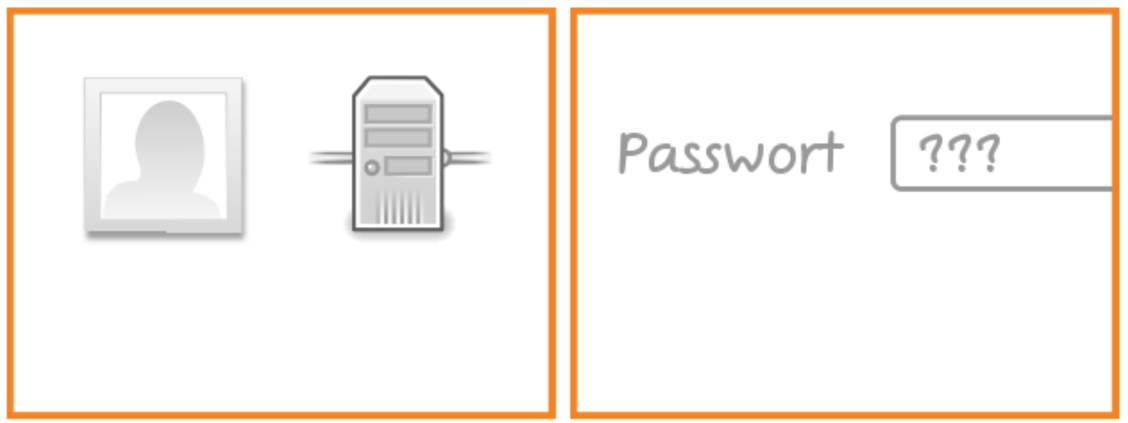
BeOne

Stefan Schlott, BeOne Stuttgart GmbH
Java-Entwickler, Scala-Enthusiast, Linux-Jünger

Seit jeher begeistert für Security und Privacy

Themenbereich - worum es gehen wird





Loginformular und Anmeldemechanismus

Es gibt verschiedene Anmeldemechanismen!

Verschiedene Möglichkeiten der Anmeldung sind mit Browser-Bordmitteln realisierbar:

- Formular
- (http auth)
- SSL-Clientzertifikat

...und natürlich noch diverse Single-Sign-On-Lösungen (z.B. OpenID, BrowserID, Facebook/Google login) mit „Dritt-Server-Unterstützung“

Der Klassiker: Login-Formular

Übertragen der Formulardaten via https

Wichtig: Formular bereits auf einer https-Seite

- Form-POST von https nach http: Browser-Warnung
- Sonst keine Gewißheit für Benutzer, dass Daten sicher übertragen werden

Beispiel: Abruf der Login-Seite via http

Server sendet:

```
<form action="https://my.site/login" method="post">
```

Be1ne

Beispiel: Abruf der Login-Seite via http

Server sendet:

```
<form action="https://my.site/login" method="post">
```

Client empfängt manipulierte Seite:

```
<form action="http://my.site/login" method="post">
```

Weitere Problematik: Eingabe in Adressleiste ohne Protokoll

```
<Virtualhost *:80>
ServerName my.site
Redirect permanent / https://my.site/
</VirtualHost>
```

Be1ne

Weitere Problematik: Eingabe in Adressleiste ohne Protokoll

```
<Virtualhost *:80>
ServerName my.site
Redirect permanent / https://my.site/
</VirtualHost>
```

Bereits dieser Redirect kann umgeschrieben werden!

Http Strict Transport Security (HSTS) als Hilfe gegen SSL Stripping

Be1ne

Was tun gegen systematisches Ausprobieren?

Bruteforce

Was tun gegen systematisches Ausprobieren?

Typischerweise keine gute Idee:

- Account sperren: Verwaltungsaufwand, DoS
- Rate Limit pro IP: Problem für Proxy-Benutzer

Bruteforce

Was tun gegen systematisches Ausprobieren?

Typischerweise keine gute Idee:

- Account sperren: Verwaltungsaufwand, DoS
- Rate Limit pro IP: Problem für Proxy-Benutzer

Rate Limit pro Username

Captcha ab einer gewissen Versuchszahl

...und natürlich gute Passwörter

Passwort-Vorgaben

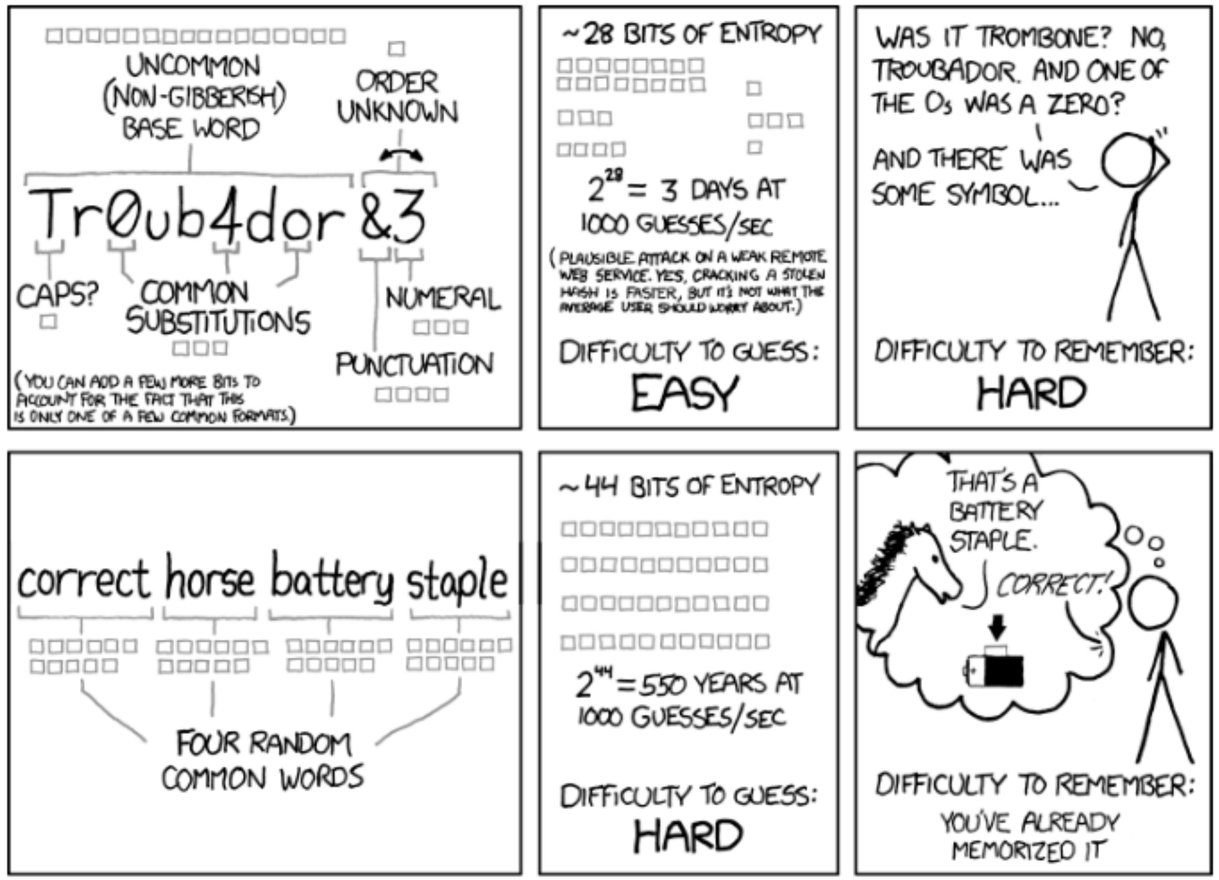
Be1ne

Mindestlänge, Mindestkomplexität haben gewissen Wert

„Gängelung“ durch zu häufiges Wechseln: Kein Sicherheitsgewinn

Passwort-Knacker auf gängige Schemata vorbereitet

Hinweise für gute Passwortwahl geben!



THROUGH 20 YEARS OF EFFORT, WE'VE SUCCESSFULLY TRAINED EVERYONE TO USE PASSWORDS THAT ARE HARD FOR HUMANS TO REMEMBER, BUT EASY FOR COMPUTERS TO GUESS.

Tipps für gute Passwörter

Be1ne

Lieber längere Passwörter als Sonderzeichen verwenden

Passwörter nicht mehrfach verwenden, Hinweis auf Generatoren mit Masterpasswort wie z.B. Password Hasher

Brauchbare Schemata für Merkhilfen:

- „correct horse battery staple“-Methode
- Anfangsbuchstaben der Wörter eines Nonesense-Satzes: „Meine Tante Martha verputzt zum Frühstück 7 Rollmöpse mit Senf, aber keinen Kaffee“ ⇒ MTMvzF7RmS,akK

Be1ne

Das Passwort als einzige Hürde?

Be1ne

Das Passwort als einzige Hürde?

2-Faktor-Authentisierung: Wissen + Besitzen

Shouldersurfing

Be1ne

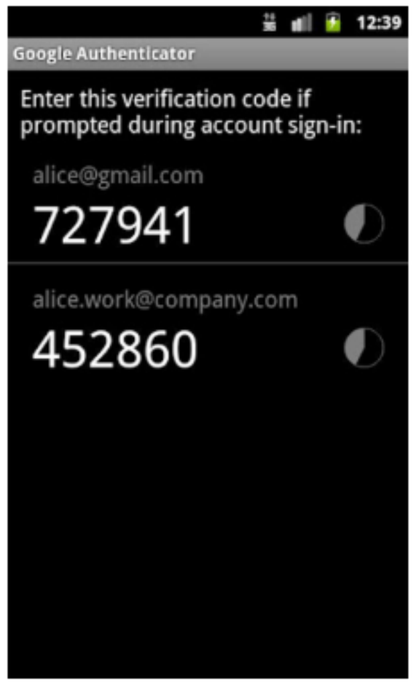
Das Passwort als einzige Hürde?

2-Faktor-Authentisierung: Wissen + Besitzen

Google Authenticator! (bzw. einen seiner Forks)

Standard: RFC6238 (Time-based OTP)

Apps für Android, iOS, Blackberry, Windows Phone, …



Shouldersurfing

Be1ne

Das Passwort als einzige Hürde?

2-Faktor-Authentisierung: Wissen + Besitzen

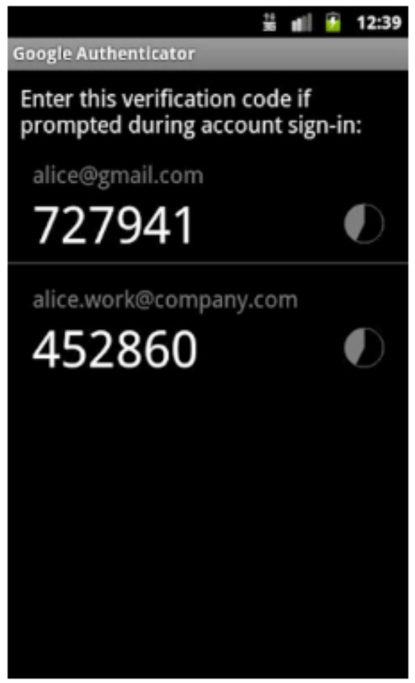
Google Authenticator! (bzw. einen seiner Forks)

Standard: RFC6238 (Time-based OTP)

Apps für Android, iOS, Blackberry, Windows Phone, …

Nicht vergessen: „Notfall-Keys“ für Notzugang ohne Smartphone

Zur Einrichtung: Probeeingabe(n) verlangen



SSL Clientzertifikate

Funktion: Symmetrisch zum SSL-Serverzertifikat

Authentisierung, wenn

- Zertifikat von konfigurierter CA signiert
- Nicht abgelaufen
- Nicht in Revocation List

SSL Clientzertifikate

Funktion: Symmetrisch zum SSL-Serverzertifikat

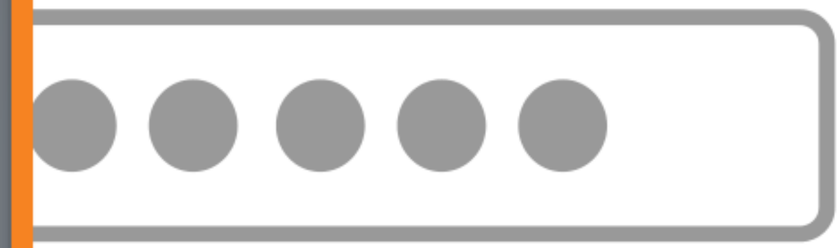
Authentisierung, wenn

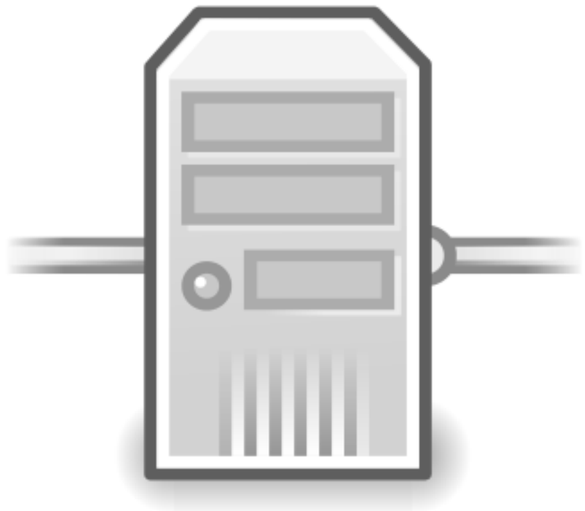
- Zertifikat von konfigurierter CA signiert
- Nicht abgelaufen
- Nicht in Revocation List

Überprüfung geschieht im SSL-Offloader bzw. SSL-Bibliothek

Möglichkeit, die Anwendung komplett nach außen abzuschotten

Selbe CA in mehreren Anwendungen: Single-Sign-On!





Passwörter speichern

Des Nutzers Kronjuwelen...

Bisherige Betrachtung:

- Das System ist sicher
- Der Benutzer verwendet überall unterschiedliche Passwörter

Des Nutzers Kronjuwelen...

Bisherige Betrachtung:

- Das System ist sicher
- Der Benutzer verwendet überall unterschiedliche Passwörter

Bisher: Passwort-Sicherheit im Kontext Web-Login

- Bandbreite limitiert Anzahl Versuche/Sek
- Möglichkeit der Einflußnahme (Captcha, ...)

Passwörter speichern

Worst-Case-Vorbereitung: Angreifer hat Zugriff aufs System. Nutzer schützen!

Passwörter nie im Klartext!

Passwörter nie reversibel verschlüsselt!

Lösung: Hashes („Fingerabdruck“ des Passworts)

Passwörter speichern

Be1ne

Worst-Case-Vorbereitung: Angreifer hat Zugriff aufs System. Nutzer schützen!

Passwörter nie im Klartext!

Passwörter nie reversibel verschlüsselt!

Lösung: Hashes („Fingerabdruck“ des Passworts)

Aber bitte nicht so:

```
$result = mysql_query(
  "SELECT * FROM users " .
  " WHERE SHA1(username) = SHA1('" . $_REQUEST["username"] . "') " .
  " AND SHA1(password) = SHA1('" . $_REQUEST["password"] . "')");
```

Kryptographischer Hash („Falltür-Funktion“)

- Einfach zu bestimmen
- Rückrichtung (passende Eingabe, die zu Hashwert X führt): Schwierig

Unschön: Identische Passwörter sind so erkennbar

Einfache Berechnung: Schnelles Durchprobieren möglich

- Rainbow Tables
- Implementierung für GPUs

Hashfunktionen

Kryptographischer Hash („Falltür-Funktion“)

- Einfach zu bestimmen
- Rückrichtung (passende Eingabe, die zu Hashwert X führt): Schwierig

Unschön: Identische Passwörter sind so erkennbar

Einfache Berechnung: Schnelles Durchprobieren möglich

- Rainbow Tables
- Implementierung für GPUs
- Google: bfb0f76744e188dda17bb62a22920c95

Hashfunktionen

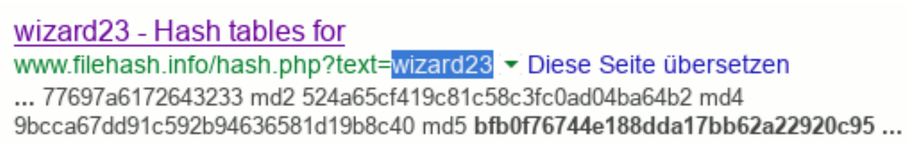
Kryptographischer Hash („Falltür-Funktion“)

- Einfach zu bestimmen
- Rückrichtung (passende Eingabe, die zu Hashwert X führt): Schwierig

Unschön: Identische Passwörter sind so erkennbar

Einfache Berechnung: Schnelles Durchprobieren möglich

- Rainbow Tables
- Implementierung für GPUs
- Google: bfb0f76744e188dda17bb62a22920c95



Hashes und Salt

Salt-Wert: Zeichenfolge, die dem Passwort hinzugefügt wird:

Statt Hash(pass): Hash(salt + pass)

Salt-Wert mitspeichern

- Identisches Passwort erzeugt nun unterschiedliche Hashes
- Vorberechnen (Rainbow Tables) schwieriger

Salt-Wert muss nicht zwingend zufällig sein; idealerweise jeden Salt-Wert nur 1x verwenden

Pepper und Garlic

Pepper: Weiterer, seiteneinheitlicher „Salt-Wert“

Nicht in Datenbank gespeichert

Nutzen umstritten. Wenn verwenden, dann HMAC-Funktion benutzen

Garlic: „New kid on the block“

Parameter, um Speicherbedarf der Hashfunktion zu erhöhen

Abwehr gegen GPU-Berechnungen

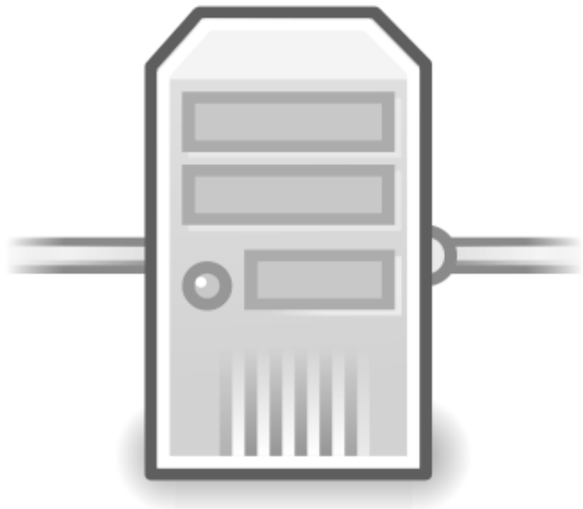
Weiteres Ausbremsen des Angreifers

Be1ne

Langsame Hashfunktionen verwenden

Also scrypt/bcrypt, nicht SHA256, SHA512, ... direkt

Mithalten mit Wettrüsten: Nötige Rechenzeit erhöhen



Session-Management

Klassisches Session-Management

Wiedererkennen eines Users über Requests hinweg

Generieren einer Session-ID

Session-ID wird bei jedem Request mitgeliefert (typischerweise Cookie)

Serverseitig: Speicher, der mit Session-ID assoziiert ist, enthält eigentliche Session-Informationen

Klassisches Session-Management

Wiedererkennen eines Users über Requests hinweg

Generieren einer Session-ID

Session-ID wird bei jedem Request mitgeliefert (typischerweise Cookie)

Serverseitig: Speicher, der mit Session-ID assoziiert ist, enthält eigentliche Session-Informationen

...unter anderem User-ID, Privilegien, ...

Für die Dauer der Session: Session-Cookie ist Username-Passwort-Äquivalent!

Session-ID prediction

Erzeugen der Session-ID: Containerabhängig

Wenn Session-ID vorhersagbar: „Einstieg“ in fremde Sessions möglich

Session-ID prediction

Erzeugen der Session-ID: Containerabhängig

Wenn Session-ID vorhersagbar: „Einstieg“ in fremde Sessions möglich

Gegenmaßnahmen:

- Überprüfen der Container-Konfiguration: Guten Zufall benutzen
- Binden der Session-ID an bestimmte IP-Adresse?

Session Fixation Attack

Be1ne

Ziel: Opfer eine dem Angreifer bekannte Session „unterschieben“

Voraussetzung: Session-ID ist von außen setzbar - z.B. Tomcats Fallback bei fehlenden Cookies:

```
http://my.site/some/url;jsessionid=...
```

Session Fixation Attack

Ziel: Opfer eine dem Angreifer bekannte Session „unterschieben“

Voraussetzung: Session-ID ist von außen setzbar - z.B. Tomcats Fallback bei fehlenden Cookies:

`http://my.site/some/url;jsessionid=...`

Angreifer generiert eine Session, lockt Opfer auf Seite z.B. mittels präpariertem Link

Session Fixation Attack

Ziel: Opfer eine dem Angreifer bekannte Session „unterschieben“

Voraussetzung: Session-ID ist von außen setzbar - z.B. Tomcats Fallback bei fehlenden Cookies:

`http://my.site/some/url;jsessionid=...`

Angreifer generiert eine Session, lockt Opfer auf Seite z.B. mittels präpariertem Link

Gegenmaßnahmen:

- Binden der Session-ID an bestimmte IP-Adresse?
- Deaktivieren solcher Funktionen, z.B. in Tomcats web.xml:

```
<session-config>
  <tracking-mode>COOKIE</tracking-mode>
</session-config>
```

Session-ID vor Angriffen schützen

Be1ne

Session-ID vor Angriffen schützen

Erschweren des Stehlens der Session-ID durch XSS-Angriffe:
Cookie-Attribut HttpOnly In Tomcats web.xml:

```
<session-config>
  <cookie-config>
    <http-only>true</http-only>
  </cookie-config>
</session-config>
```

Sichern der Session-ID

Be1ne

Session-ID vor Angriffen schützen

Erschweren des Stehlens der Session-ID durch XSS-Angriffe:
Cookie-Attribut HttpOnly In Tomcats web.xml:

```
<session-config>
  <cookie-config>
    <http-only>true</http-only>
  </cookie-config>
</session-config>
```

Wird https verwendet: Übertragung via http mittels Secure unterbinden. In Tomcats web.xml:

```
<session-config>
  <cookie-config>
    <secure>true</secure>
  </cookie-config>
</session-config>
```

Mehrfach-Logins

Be1ne

Sollen mehrere gleichzeitig gültige Session-Cookies möglich sein?

Vorteil: Auf mehreren Browsern angemeldet bleiben

Nachteil: Diebstahls-Risiko, vergessen abzumelden, etc.

Mehrfach-Logins

Be1ne

Sollen mehrere gleichzeitig gültige Session-Cookies möglich sein?

Vorteil: Auf mehreren Browsern angemeldet bleiben

Nachteil: Diebstahls-Risiko, vergessen abzumelden, etc.

Bei naiver Verwendung der Sessions: Mehrfachlogin möglich

Hinweis: Mögliche inkonsistente Daten bei Verwendung von Session als Daten-Cache

Mehrfach-Logins

Be1ne

Sollen mehrere gleichzeitig gültige Session-Cookies möglich sein?

Vorteil: Auf mehreren Browsern angemeldet bleiben

Nachteil: Diebstahls-Risiko, vergessen abzumelden, etc.

Bei naiver Verwendung der Sessions: Mehrfachlogin möglich

Hinweis: Mögliche inkonsistente Daten bei Verwendung von Session als Daten-Cache

Best Practice:

- Übersicht über gültige Sessions (samt letztem Zugriff)
- Ausloggen über Webseite (einzeln oder alle)

Passwort ???

Password Recovery

Vergessene Passwörter

Be1ne

Fragen nach Lieblingstier, Name der Mutter, etc. leicht zu erraten

Recovery per E-Mail

Vergessene Passwörter

Be1ne

Fragen nach Lieblingstier, Name der Mutter, etc. leicht zu erraten

Recovery per E-Mail

- Nie Klartext-Passwörter per Mail
- Einmal-Links mit begrenzter Lebenszeit, ggfs. auf IP-Adresse beschränkt
- Problem: E-Mail als Single Point of Failure

Vergessene Passwörter und Two-Factor-Authentication

Authentisierungs-Token verloren?

Einloggen mittels „Notfall-Keys“, Re-Keying

Vergessene Passwörter und Two-Factor-Authentication

Be1ne

Authentisierungs-Token verloren?

Einloggen mittels „Notfall-Keys“, Re-Keying

Passwort vergessen?

Recovery-Link plus Token-Abfrage

Vergessene Passwörter und Two-Factor-Authentication

Be1ne

Authentisierungs-Token verloren?

Einloggen mittels „Notfall-Keys“, Re-Keying

Passwort vergessen?

Recovery-Link plus Token-Abfrage

→ Kein Single Point of Failure!

OAuth & Co.

„Ersatzpasswort“ für Apps

Häufig gewünscht: API-Zugang, Mashups, etc.

User-Passwort in fremde Apps eingeben: Schlecht

- Liegt dort im Klartext
- Passwort ändern wird aufwendig

„Ersatzpasswort“ für Apps

Häufig gewünscht: API-Zugang, Mashups, etc.

User-Passwort in fremde Apps eingeben: Schlecht

- Liegt dort im Klartext
- Passwort ändern wird aufwendig

Standard-Procedere: OAuth, alternativ: Generierte Tokens

Separater Schlüssel für Apps

Anwendung kann Schlüsseln unterschiedliche Rechte einräumen

Tokens vs. Passwort-Wechsel

Vorsicht: User-Passwort-Wechsel macht OAuth-Tokens *nicht* ungültig

Best Practice:

- Liste mit vergebenen Tokens (samt Rechten)
- Möglichkeit, einzelne Tokens zurückzuziehen
- Hinweis bei erfolgreicher Passwort-Änderung, dass verknüpfte Apps nach wie vor Zugriff haben

Fazit

Security-Themen können tricky sein

Passwörter: Ein Konzept, von dem wir uns mittelfristig verabschieden müssen

Beim Thema Login und Session-Management: Wenig „Baukästen“, Defaultkonfiguration überprüfen

Strebe nach „Secure By Default“

Preparedness, Constant Vigilance

Be1ne
stuttgart

Dr. Stefan Schlott
http://www.beone-group.com/
stefan.schlott@beone-group.com
Twitter: @_skyr

Bildquellen

- Password strength (http://xkcd.com/936/) (CC) BY-NC Randall Munroe