HACK YOUR OWN WEBSITE!

WEB SECURITY IM SELBSTVERSUCH

Dr. Stefan Schlott, BeOne Stuttgart GmbH

ABOUT.TXT

Stefan Schlott, BeOne Stuttgart GmbH

Java-Entwickler, Scala-Enthusiast, Linux-Jünger

Seit jeher begeistert für Security und Privacy

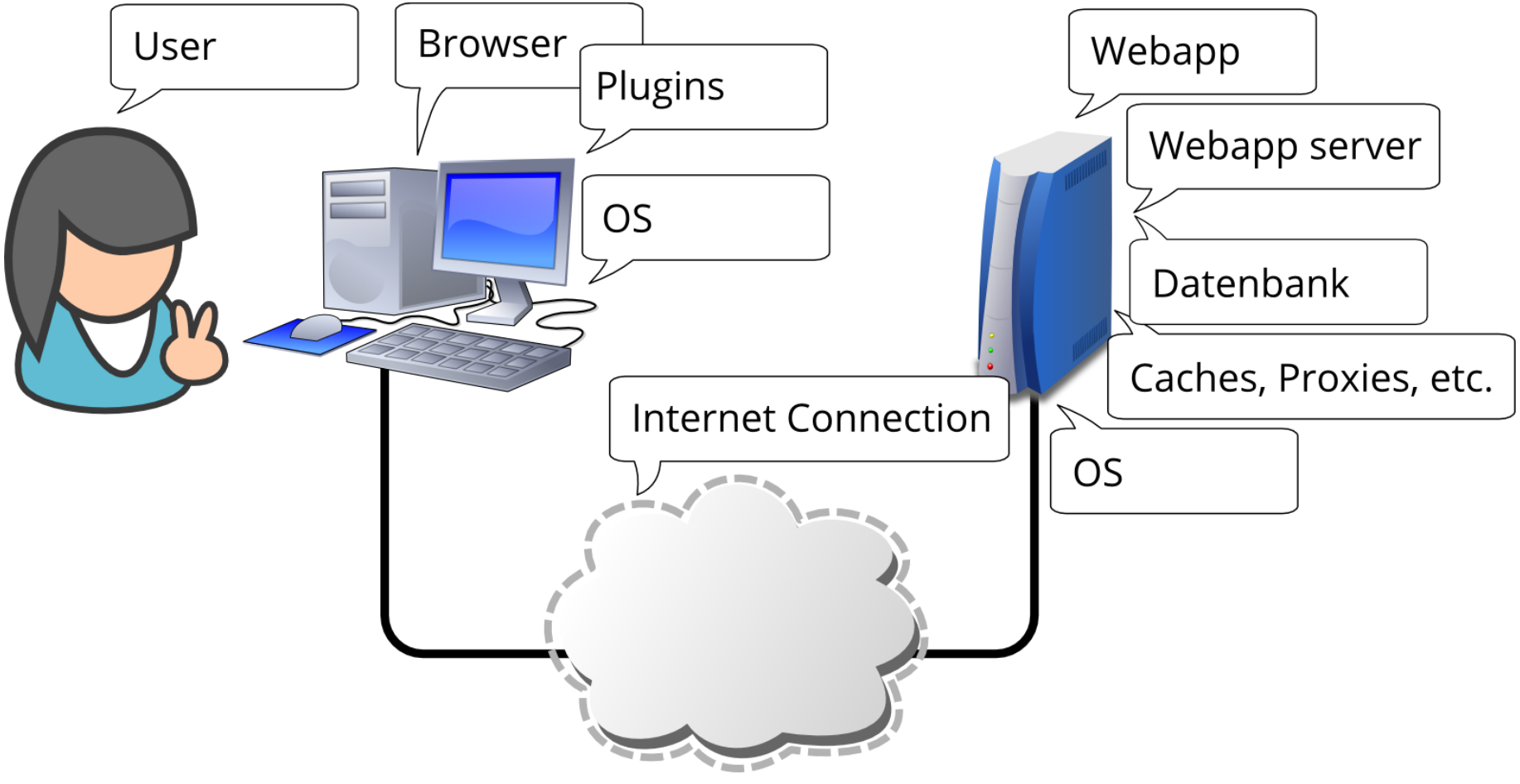
DIE OWASP TOP-10

Alpha und Omega der Security-Talks :-)

TOP 10 VON 2013

1. Injection
2. Broken Authentication and Session Management (von 3)
3. Cross-Site Scripting (XSS) (von 2)
4. Insecure Direct Object References
5. Security Misconfiguration (von 6)
6. Sensitive Data Exposure (von 7 und 9)
7. Missing Function Level Access Control (von 8)
8. Cross-Site Request Forgery (von 5)
9. Using Known Vulnerable Components (von 6)
10. Unvalidated Redirects and Forwards

ANGRIFFSZIELE



MAL DEN „SCHURKE“ SPIELEN ;-)

WERKZEUGE

Tools und willige Opfer

KALI LINUX

Kali Linux Penetration Testing Distro

Umfangreiche Tool-Sammlung

Livesystem oder Installation

Empfehlung: In VM installieren

TESTANWENDUNGEN

Erste Übungen an absichtlich schwache Anwendungen:

- OWASP WebGoat
- DVWA (Damn Vulnerable Web App)
- The BodgeIt Store
- …

Niemals direkt am Netz laufen lassen!

Empfehlung: Zweite VM (teils Live-CD verfügbar)

ERSTKONTAKT

OWASP ZED ATTACK PROXY (ZAP)

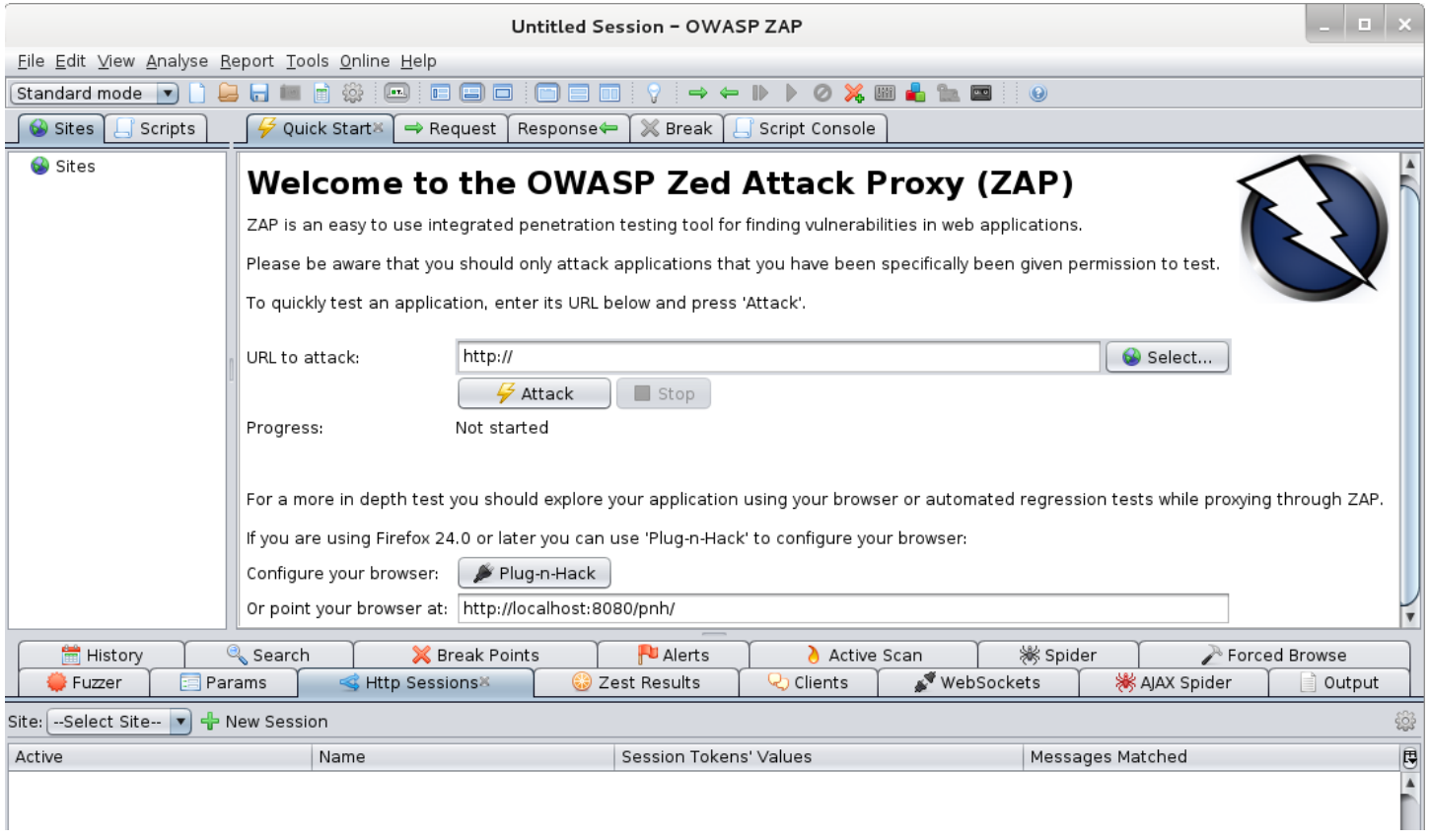
„Schweizer Taschenmesser“ für Experimente im Browser

Beobachten, Abfangen und Manipulieren von Browseranfragen (SSL-Man-in-the-Middle)

Passiver Scanner:
Erkennung möglicher Probleme in Anfragen

Spider, aktiver Scanner:
Systematische Suche nach Standardproblemen

EINRICHTUNG: HACK'N'PLAY



Automatische Konfiguration via Browser-Plugin

(Button „redet“ mit Default-Browser)

BETRACHTEN UND ÄNDERN VON REQUESTS

Requests werden in hierarchischen Baum festgehalten

Chronologisch: History-Tab

Mittels  : Alle Request abfangen

Bestimmte Requests mit Breakpoints  abfangen

Eigene Anfragen mittels Manual Request Editor (Tools)

SESSION MANAGEMENT

Erkennt typische Sessioncookies (über  anpassbar)

Durch Aktivieren: Umschalten zwischen Sessions
(ZAP ersetzt Sessioncookies vom Browser
durch die gespeicherten)

TYPISCHES VORGEHEN

Manuelles Browsen

Festlegen des Kontexts, Ausschluß der Logout-Seite

Spider starten

Ggfs. Forced Browsing (typische URLs probieren)

Ggfs. Active Scan

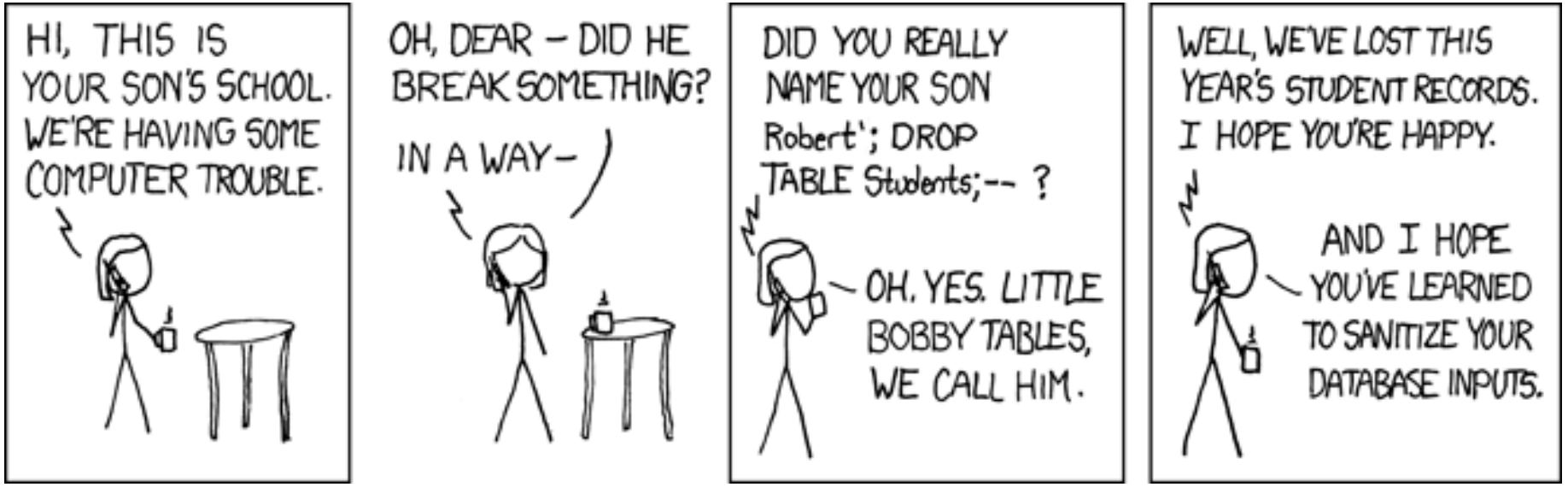
SQL INJECTION

Ist der kleine Bobby Tables zu Hause?

SQL INJECTION 101

SQL-Statement wird direkt zusammengebaut

```
name = … // (direkt aus dem Webformular)
result = SQL.exec("SELECT * FROM Students WHERE NAME='" + name + "';");
```




```
SELECT * FROM Students WHERE NAME='Robert';
DROP TABLE Students;--';
```

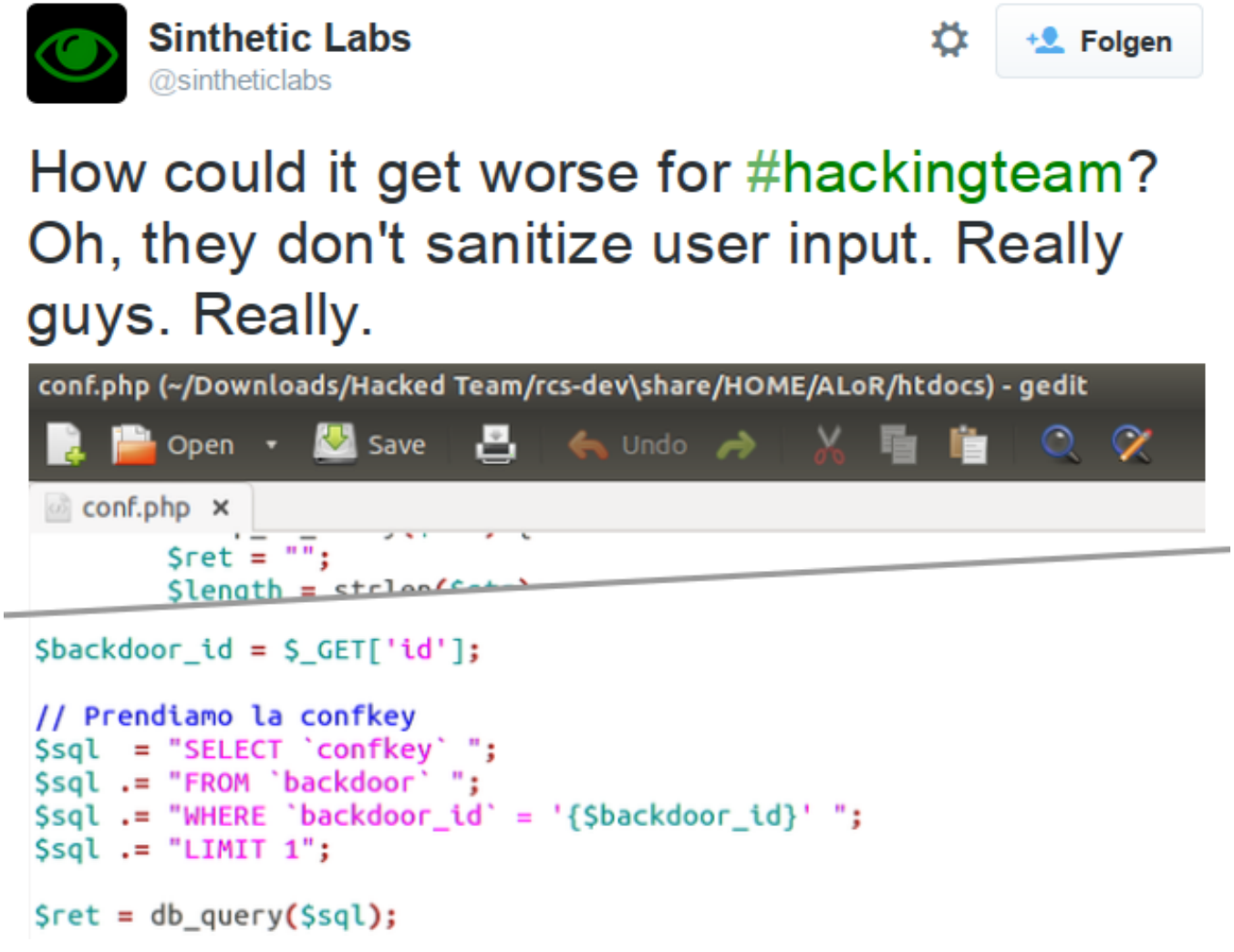
VERMUTUNG

Demo: WebGoat - Injection Flaws

Vermutung: Prüfung der Anmeldung durch

```
sql("SELECT COUNT(*) FROM users WHERE
  NAME='" + $name + "' AND PASSWORD='" + $pass + "';")
```

...PASSIERT AUCH „PROFIS“



BLIND SQL INJECTION

Erkennen der Anfälligkeit…

- …wenn nur eine generische Fehlermeldung erscheint
- …nur eine leere Seite erscheint oder gar keine Verhaltensänderung erkennbar ist

Interpretation des geänderten Verhaltens

Timing-basierte Erkennung: Versuch, die Datenbankantwort meßbar (damit: erkennbar) zu verzögern

SQLMAP

Systematisches Durchprobieren verschiedener SQL-Injection-Techniken

Datenbank-Fingerprinting

Datenbank-Dump

Und vieles mehr…

SQLMAP

```
sqlmap -u 'http://…/vulnerabilities/sqli/?id=1&Submit=Submit'
--cookie 'security=low; PHPSESSID=…' -p id
```

- -u URL
- -p zu untersuchender Parameter

Einige weitere Optionen:

- -f -b Weitere Systeminfos via Fingerprinting
- --tables Tabellenschema
- --dump Daten-Dump

KORREKTES ANWENDUNGSVERHALTEN

Saubere Validierung der Eingaben
(nicht schädlich für eigenes Programm)

Encodierung der Ausgaben an andere Anwendungen (hier:
Escaping von für SQL gefährlichen Zeichen)

Für SQL: Prepared Statements, Serialisierungs-Bibliothek

SQL IST NICHT ALLES!

Injection ist kein SQL-spezifisches Problem!

- Zusammenbauen von JPQL-Strings
- JSP Expression Language
- Klasseninstantiierung durch Klassenangaben in YAML
- …

CROSS SITE SCRIPTING

Injection im Browser

CROSS SITE SCRIPTING

Server sendet Daten an Client, die dort zur Ausführung kommen (üblicherweise: JS)

Spring-Beispiel: URL-Parameter via spring:message:

```
<spring:message text="${param['info']}"></spring:message>
```

JSP EL (sofern das Attribut `variable` nicht vorher behandelt wurde):

```
${variable}
```

QUELLEN FÜR XSS-PROBLEME

Persistent XSS: Daten auf Server hinterlegt

Reflected (non-persistent) XSS: Nicht durch Daten auf Server, sondern durch präparierten Link o.ä.

Anzeige von URL-Parametern oder Cookies

Einfügen von Werten aus der Datenbank

Verwenden von Werten aus Web-APIs

Anzeige von hochgeladenen Dateien

ERKENNUNG MIT ZAP

ZAPs „Active Scan“ versucht, XSS zu erkennen

Heuristiken versuchen, manipulierte Eingaben in der Ausgabe zu finden

Ausführlichere Tests: ZAP Fuzzer

In Request zu manipulierenden String markieren, Fuzzer im Kontextmenü wählen

Verschiedene Regel-Sets verfügbar

XSS IST NICHT HARMLOS!

Erhebliche Eingiffsmöglichkeiten (Beispiel: beef)

Präparierte Webseite kann „Sprungbrett“ ins Intranet sein

Kompromittierung anderer Webanwendung in selber Cookie-Domain

KORREKTES ANWENDUNGSVERHALTEN

Encodierung der html-Ausgabe (Escaping für html)

…entweder durch umsichtigen Programmierer

…oder durch geeignete Template-Frameworks für Ausgabe (zu bevorzugen)

LOGIN BRUTE FORCE

…freundliches Anklopfen geht anders!

HYDRA

Durchprobieren von Username-Passwort-Paaren für verschiedenste Protokolle

Eingabe: Username- und Passwortliste

Zusätzlich: Einfache Varianten des Usernamens als Passwort verwenden

LOGIN MIT HTTP GET

```
hydra -L users.txt -P passwords.txt …host… http-get-form
'/vulnerabilities/brute/
:username=^USER^&password=^PASS^&Login=Login#
:Username and/or password incorrect
:H=Cookie: PHPSESSID=…'
```

Modul `http-get-form` benötigt drei Parameter:

- URL-Pfad
- URL-Parameter mit Platzhaltern für User und Passwort
- Charakteristischen Text in der Antwort, der einen Fehlschlag anzeigt
- Optional: Weitere http-Header

KORREKTES ANWENDUNGS-VERHALTEN

Zu häufige Fehleingaben sollten ausgebremst werden

Ausbremsen z.B. durch captchas, nicht Aussperren!

Regeln im Idealfall pro Username

Für interne Anwendungen mit wenig Usern:

Aussperren eine valide Option

Nachrüstung durch den Admin mittels fail2ban
(Voraussetzung: Fehlgeschlagenes Login in Logs erkennbar)

FAZIT

Experimentieren lohnt sich - und macht Spaß!

Automatisierte Scanner helfen, ersetzen aber keinen manuellen Scan

Testanwendungen nur lokal erreichbar installieren

BeOne stuttgart

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

BILDQUELLEN

- Exploits of a mum (CC) BY-NC Randall Munroe