

# Varnost in PHP

Najbolj pogoste varnostne luknje  
in primeri napadov

Gašper Kozak  
<?php konferenca 2009

# Kdo

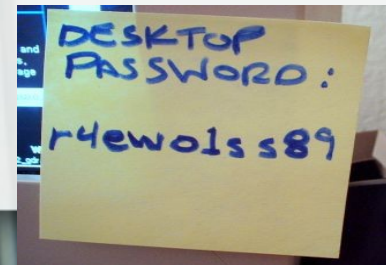
- Gašper Kozak, 30 let
- programiram 15 let, 9 od tega za splet
- razvijalec v podjetju Tobonet, kjer razvijamo oglaševalski sistem Httpool AdPlatform
- v prostem času: Widelimage, varnost in še kaj
- aktiven član spletnih skupnosti
  - php-si, pehape.si, sitepoint, devnetworks, stack overflow, blogi, ...
- ne hekam :)

# Oris delavnice

- nivoji varnosti in malo zgodovine
- safe\_mode in open\_basedir
- varnostni problemi, primeri in rešitve
  - Luknja™
  - sproti poizkušajte na svojih straneh :)
- povzetek in zaključek
- pa še:
  - nekatere stvari bom večkrat ponovil
  - slovenščina je toga

# Nivoji varnosti

- fizična varnost (fizični dostop)
- sistem in okolje (OS, web strežnik, php.ini)
- nižje-nivojski aplikacijski nivo:
  - razumevanje prepleta tehnologij
  - preverjanje podatkov
  - priprava za vnos v bazo, html, mail, ...
- višje-nivojski aplikacijski nivo (captcha, dovoljenja, napačne prijave, ...)
- uporabniški nivo (post-it listki z gesli ftw!)



# Zakaj je PHP nevaren?

- zgodovina: static html, cgi programi, ... PHP znotraj HTML! Neznosna lahkost spletnega programiranja.
- zmogljiv in enostaven ...
- ... tudi na račun varnosti: magic\_quotes, url include, register\_globals, ...
- orodje je nevarno toliko, kolikor ga pravilno uporabljaš
- hvala, phpBB in PHPNuke! (in še kdo)

# safe\_mode

- omejevanje dostopa do datotek iz PHP-ja na lastnika izvršujoče skripte
- mnoge razširitve in funkcije tega ne uporabljajo
- PHP datoteke piše z web-server uporabnikom
  - niso berljive s skriptami, posnetimi preko FTP
  - script-uploader script
  - tako uploadana skripta ima več dostopa
- dostop z drugimi orodji ni omejen
- arhitekturno napačen in odstranjen v PHP6

# open\_basedir

- omeji dostop do datotek na določen direktorij in drevo pod njim
- se preverja pri vsakem dostopu do datoteke
- se jo lahko nastavi za posameznega uporabnika
- bolj učinkovita rešitev od `safe_mode`

# register\_globals: problem

- spremenljivke iz GPSCE avtomatično na voljo v skripti
- programerjem naj bi olajšala programiranje
- od kje spremenljivka pride?
- pretežno iztrebljena luknja:
  - privzeto ugasnjena nastavitvev
  - kmalu je ne bo več (PHP 6)

# register\_globals: rešitev

- spremenljivke je treba vedno prednastaviti (po slovensko: inicializirati)
- `register_globals = Off`

# include exploit: problem

- omogoča vključitev poljubne kode v izvajanje strežniške PHP datoteke preko include/require:
  - z zunanjega strežnika (`allow_url_include`)
  - lokalno (`upload`, `log exploit`)
- zelo nevarna luknja

# include exploit: rešitev

- `allow_url_include = Off`
  - že nekaj časa ločen od `allow_url_fopen`
  - samo po sebi to ni dovolj
- preverjanje parametrov za nedovoljene znake
  - `.. /` in znaki pod `0x20 = #32 = space`
- preverjanje uploadanih datotek
  - preverjanje končnice ni dovolj
  - `FileInfo?`
- privilegiji web-server uporabnika

# filesystem access: problem

- podobno remote include luknji, samo da ni include, temveč branje datoteke
- skripta, ki servira datoteko s strežnika, omogoča branje poljubnih datotek na strežniku
  - omejeno na tiste, za katere ima web-strežnik dovolj dovoljenj

# filesystem access: rešitev

- preverjanje parametrov
  - nedovoljeni znaki za relativno pot .. / : \
  - posebni znaki < 0x20
  - preverjena osnovna pot, iz katere skripta lahko bere (realpath)
  - končnica datotek
- privilegiji web-strežnik uporabnika

# e-mail header injection: problem

- enostavno pošiljanje e-maila:
  - `mail($_GET['recipient'], $_GET['subject'], $_GET['message'], "From: " . $_GET['from']);`
- napadalec poda vrednosti, ki spremenijo ali dodajo nove headerje v e-mail sporočilo
- spam
- ponarejanje izvora

# e-mail header injection: rešitev

- deloma zakrpano s strani PHP-ja (To, Subject)
- preveriti vse vhodne podatke
  - predvsem e-mail naslove
  - From, Subject in To ne smejo vsebovati \r ali \n
  - pravzaprav nobenega posebnega znaka < 0x20
- za pošiljanje mailov uporabite PHPMailer, SwiftMailer, PEAR::Mail, ...

# SQL injection – napad z narekovaji: problem

- napad z vrivanjem kode v poizvedbo
- primer:
  - `select * from user where username = '{$_GET['uname']}' and password = '{$_GET['pass']}'`
  - `login.php?uname=admin' -- &pass=`
  - `select * from user where username = 'admin' -- 'and password = "`

# SQL injection – napad z narekovaji: rešitev?

- brisanje znakov ' in "?
  - je zaščita, ampak poslabša uporabniško izkušnjo
- magic\_quotes\_gpc?
  - ne gredo vsi podatki v bazo
  - v bazo gredo tudi podatki, ki niso iz GPC
  - \ ni escape znak za vse baze
  - kmalu bo ukinjen
- addslashes?
  - \ ni escape znak za vse baze ...

# SQL injection – napad z narekovaji: rešitev

- `mysql_real_escape_string`
  - oz. pač glede na bazo
- še bolje: prepared statements
  - PDO
- ORM (Doctrine, Propel, EZPDO, ...)
- frameworki

# SQL injection – napad brez narekovajev: problem

- podatki, ki ne vsebujejo narekovajev
  - gredo mimo magic\_quotes, addslashes, mysql\_real\_escape\_string, ...
- napad na numerična polja
  - delete from novica where id = \$\_GET['id']
  - [delete.php?id=235 or 1 = 1](#)
  - delete from novica where id = 235 or 1 = 1
  - delete from novica where true

# SQL injection – napad brez narekovajev: rešitev

- preverjanje (validation)
  - is\_numeric, is\_float, preg\_match
  - ctype, filter
- normalizacija (sanitization)
  - intval, floatval
- razlika med preverjanjem in normalizacijo
  - **id=25abc** ni pravilen parameter
  - is\_numeric = false, intval = 25
- frameworki

# XSS: problem

- XSS = cross-site scripting = napad z vstavljanjem kode v spletno stran
- izkoriščanje nepravilno pripravljenih podatkov, ki gredo iz programa v HTML/JavaScript
- vstavljena koda se prikaže in izvede s privilegiji napadenega

# XSS: rešitev

- preverjanje **vseh** parametrov
  - brisanje `<script>` tagov ali znakov `<` `>` ni rešitev
  - whitelist `>` blacklist
- `htmlentities` ali `htmlspecialchars` na **vseh** izhodnih spremenljivkah
- spodobni frameworki in nekatera (ne vsa!) templating orodja imajo to že vgrajeno
  - ne izklaplajte (čeprav se da)

# CSRF: problem

- CSRF = cross-site request forgery = napad z zahtevo preko druge strani:
  - napadalec sestavi URL veljavne akcije:  
[http://stran.com/user\\_del?id=3](http://stran.com/user_del?id=3)
  - ta URL postavi na stran B  
``
  - žrtev obišče stran B
  - brskalnik naredi zahtevo na stran A
- phpMyAdmin: brisanje tabel
- Shiflett vs. Amazon

# CSRF: rešitev

- pravilna uporaba HTTP protokola: POST za spreminjanje, GET za branje
  - malo pomaga, ampak ne reši
- preverjanje referrerja
  - samo, če je pravilno izvedeno: preverjanje cele vrednosti
  - nerodno, če lahko kličemo iste akcije z več vstopnih točk
  - referrer ni dovolj zanesljiv
- form tokens (žetoni): komplicirano, zanesljivo

# session hijacking: problem

- session hijacking = kraja, ugrabitev seje
- napadalec mora imeti dostop do session\_id-ja:
  - fizičen dostop
  - nevarni programi (virusi, črvi, ...)
  - naiven uporabnik
  - XSS
- napadalec uporabi session\_id za dostop

# session hijacking: rešitev

- preverjanje dodatnih parametrov (User Agent, IP)
  - User Agent se lahko spremeni
  - IP tudi
- ni zanesljive rešitve
  - uporabniki so varnostna luknja :)

# session fixation: problem

- napadalec vnaprej pozna ali določi session\_id in ga podtakne žrtvi
- žrtev odpre sejo s tem ID-jem in se prijavi
- ...
- napadalec: profit!

# session fixation: rešitev

- `session.use_only_cookies = true`
- `session.use_trans_sid = false`
- aplikacijska logika:
  - `session_regenerate_id()` ob vsaki spremembi privilegijev

# shell injection: problem

- nepreverjeni in nepripravljeni podatki gredo v shell
  - in tam rajajo
- izvajanje poljubnih ukazov s privilegiji web-server uporabnika

# shell injection: rešitev

- preverjanje parametrov (zveni znano?)
- `escapeshellarg()`
- `escapeshellcmd()`
- omejevanje dostopa za web-server uporabnika na sistemskem nivoju
  - delo za administratorja

# Pravilna pot podatka

- vstopni podatek:
  - GET, POST, COOKIE, REQUEST, SERVER, ENV, db, http, ftp, lokalne datoteke, ...
- po potrebi odstraniti slashe (magic\_quotes\_gpc)
- preverjanje podatkov (validation)
  - is\_numeric, strlen, preg\_match, ctype, filter ...
- normalizacija podatkov (sanitization)
  - intval, strip\_tags
- v aplikaciji so podatki v potencialno nevarni obliki
- priprava glede na cilj:
  - db, html, js, css, datoteka, shell, ...

# Povzetek

- vhodne podatke je treba vedno preveriti in normalizirati
- izhodne podatke je treba vedno pravilno pripraviti za pisanje
- = filter input, escape output
- dobri frameworki pomagajo
- aplikacija še vedno ni varna ...
  - to niso vse možne luknje
  - obstajajo še ostali štirje nivoji varnosti, ki morajo prav tako biti dobro zaščiteni

# Zaključek

Programiranje varnih aplikacij ni enostavno.

Hvala za oči in ušesa.

Vprašanja?

# Viri

- Essential PHP Security (Chris Shiflett, 2006)
- [http://en.wikipedia.org/wiki/Remote\\_File\\_Inclusion](http://en.wikipedia.org/wiki/Remote_File_Inclusion)
- <http://aymanh.com/remote-inclusion-in-php>
- <http://itbloggen.se/cs/blogs/secteam/archive/2009/01/26/alternative-ways-to-exploit-PHP-remote-file-include-vul>
- [http://en.wikipedia.org/wiki/E-mail\\_injection](http://en.wikipedia.org/wiki/E-mail_injection)
- [http://www.nyphp.org/phundamentals/email\\_header\\_injection.php](http://www.nyphp.org/phundamentals/email_header_injection.php)
- <http://shiflett.org/articles/sql-injection>
- [http://en.wikipedia.org/wiki/Sql\\_injection](http://en.wikipedia.org/wiki/Sql_injection)
- <http://shiflett.org/blog/2006/jan/addslashes-versus-mysql-real-escape-string>
- <http://xkcd.com/327/>
- <http://unixwiz.net/techtips/sql-injection.html>
- <http://si.php.net/manual/en/security.database.sql-injection.php>
- <http://shiflett.org/blog/2006/jan/addslashes-versus-mysql-real-escape-string>
- [http://en.wikipedia.org/wiki/Cross-site\\_scripting](http://en.wikipedia.org/wiki/Cross-site_scripting)
- <http://shiflett.org/blog/2005/jan/xss-cheatsheet>

# Viri

- <http://shiflett.org/articles/foiling-cross-site-attacks>
- [http://en.wikipedia.org/wiki/Cross-site\\_request\\_forgery](http://en.wikipedia.org/wiki/Cross-site_request_forgery)
- <http://shiflett.org/blog/2007/mar/my-amazon-anniversary>
- <http://shiflett.org/articles/session-fixation>
- <http://si.php.net/manual/en/session.configuration.php#ini.session.use-trans-sid>
- [http://phpsec.org/projects/phpsecinfo/tests/use\\_trans\\_sid.html](http://phpsec.org/projects/phpsecinfo/tests/use_trans_sid.html)
- [http://en.wikipedia.org/wiki/Code\\_injection](http://en.wikipedia.org/wiki/Code_injection)
- <http://si2.php.net/features.safe-mode>
- [http://ilia.ws/archives/18\\_PHPs\\_safe\\_mode\\_or\\_how\\_not\\_to\\_implement\\_security.html](http://ilia.ws/archives/18_PHPs_safe_mode_or_how_not_to_implement_security.html)
- <http://shiflett.org/articles/shared-hosting>
- in še kaj

# Avtor na spletu

- [gasper.kozak@gmail.com](mailto:gasper.kozak@gmail.com)
- <http://kozak.si/widethoughts/>
- [http://www.ohloh.net/accounts/gasper\\_k](http://www.ohloh.net/accounts/gasper_k)
- [http://sourceforge.net/users/gasper\\_k/](http://sourceforge.net/users/gasper_k/)
- <http://www.linkedin.com/in/gasperkozak>
- <http://fatg.sopca.com/>
- [http://twitter.com/gasper\\_k](http://twitter.com/gasper_k)
- [http://www.noovo.com/u/gasper\\_k/](http://www.noovo.com/u/gasper_k/)