OWASP

The Open Web Application Security Project

OWASP
The Open Web Application Security Project

Web Security

based on PHP language

امنیت نرم افزار های وب

با تمرکز بر روی زبان PHP

www.owasp.org - www.bostandoust.ir

OWASP
The Open Web Application Security Project

سعید بستان دوست

کارشناس نرم افزار و متخصص امنیت شبکه و نرم افزار

ssbostan@hackermail.com

www.bostandoust.ir

OWASP
The Open Web Application Security Project

Web Applications Vulnerabilities

بررسی آسیب پذیری های رایج در نرم افزار های وب

OWASP
The Open Web Application Security Project

File Inclusion Vulnerability

آسیب پذیری فراخوانی فایل

- عوامل آسیب پذیری: فراخوانی فایل از ورودی فیلتر نشده کاربر
- توابع آسیب پذیر: include, include_once, require, require_once, fopen, file_get_contents
- فراخوانی و اجرای فایل های PHP موجود در سرور محلی یا سرور خارجی
- دسترسی به فایل هایی که خارج از دسترس عموم هستند.
- دسترسی به پیکربندی سیستم
- تزریق کد و دسترسی به خط فرمان سرور
- کلید واژه ها: LFI, RFI, Local File Inclusion, Remote File Inclusion

How to prevent File Inclusion vulnerability?

چگونه از آسیب پذیری فراخوانی فایل جلوگیری کنیم؟

- عدم استفاده از ورودی کاربر برای فراخوانی فایل(GET, POST, REQUEST)
- در صورت نیاز به فراخوانی از ورودی کاربر، آن را بررسی و فیلتر کنید.
- تنظیمات PHP را به درستی تنظیم نمایید.
- open_basedir, allow_url_include, allow_url_fopen
- فایل های PHP با سطح کاربری محدود اجرا شوند.
- نمونه کد آسیب پذیر:

```
<?php include($_GET[“page”] . “.php”); ?>
```

OWASP
The Open Web Application Security Project

SQL Injection Vulnerability

آسیب پذیری تزریق دستورات پرس و جو پایگاه داده

- عوامل آسیب پذیری: استفاده نادرست و عدم فیلتر ورودی های کاربر در دستورات پرس و جو پایگاه داده
- توابع آسیب پذیر: هر تابعی که با پایگاه داده در ارتباط باشد. برای مثال: mysql_query
- دسترسی به داده هایی که نباید در دسترس عموم کاربران قرار گیرد.
- درج، ویرایش و حذف داده ها و اجرای دستورات مدیریتی پایگاه داده که نباید توسط کاربران عمومی اجرا شوند.
- دسترسی به فایل های پیکربندی سیستم
- استخراج داده ها یا تزریق کد های مخرب در فایل سیستم
- کلید واژه ها: SQLi, Blind SQL Injection

OWASP
The Open Web Application Security Project

How to prevent SQL Injection vulnerability?
چگونه از آسیب پذیری پایگاه داده جلوگیری کنیم؟

- از توابع سنتی **mysql** استفاده نکنید و بجای آن از **PDO** یا **mysqli** استفاده نمایید.
- تمامی داده هایی که توسط کاربر وارد می شوند را بررسی و فیلتر نمایید.
- در صورت استفاده از توابع **mysql** با تابع **mysql_real_escape_string** کاراکترهای ویژه را کد کنید.
- سطح دسترسی کاربر پایگاه داده را محدود کنید.
- دستورات پرس و جو را با دقت نوشته و از * استفاده نکنید.
- نمونه کد آسیب پذیر:

```
<?php mysql_query(“select title from posts where id=’$id’;”); ?>
```

OWASP
The Open Web Application Security Project

Code Injection Vulnerability

آسیب پذیری تزریق کدهای PHP

- عوامل آسیب پذیری: استفاده نادرست و عدم فیلتر ورودی های کاربر در توابع اجرای کد
- توابع آسیب پذیر: توابعی مانند eval و preg_replace که بدون فیلتر صحیح مورد استفاده قرار گیرند.
- اجرای دستورات زبان PHP در سمت سرور
- درج، ویرایش و حذف فایل یا استفاده از منابع سرور برای حملات توزیع شده
- دسترسی به فایل های پیکربندی سیستم
- استخراج داده ها یا تزریق کد های مخرب در فایل سیستم
- کلید واژه ها: PCI, Eval Injection, PCRE Exploit

How to prevent Code Injection vulnerability?
چگونه از آسیب پذیری تزریق کد جلوگیری کنیم؟

- از ورودی کاربر در تابع **eval** استفاده نکنید.
- در صورت نیاز به تابع **preg_replace** از امکان اجرای کد **(e)** استفاده نکنید.
- ورودی های کاربر را بررسی و فیلتر نمایید.
- سطح دسترسی **PHP** را محدود و آن را در محیط **CGI** اجرا نمایید.
- توابع حساس را توسط **disable_functions** محدود کنید.
- نمونه کد آسیب پذیر:

```
<?php echo preg_replace(“/(.*)/e”, “strtoupper(\\1)”, $input); ?>
```

OWASP
The Open Web Application Security Project

Object Injection Vulnerability

آسیب پذیری تزریق Object در PHP

- عوامل آسیب پذیری: عدم بررسی و فیلتر ورودی کاربر در بازیابی مقادیر ذخیره شده
- توابع آسیب پذیر: استفاده از تابع **unserialize** برای بازیابی مقادیری که توسط **serialize** کد شده‌اند.
- ایجاد نمونه از کلاس های طراحی شده در نرم افزار
- اجرای توابع و متدهایی که نباید توسط کاربران عادی اجرا شوند.
- به صورت ساده حملات این آسیب پذیری از متدهای __wakeup و __destruct صورت می گیرد.
- امکان اجرای انواع حملات سطح نرم افزار
- کلید واژه ها: PHP Object Injection, unserialize Exploit

How to prevent Object Injection vulnerability?

چگونه از آسیب پذیری تزریق Object جلوگیری کنیم؟

- در صورت نیاز به ذخیره مقادیر آن را در سمت کاربر ذخیره نکنید.
- برای کد نمودن مقادیر از توابع json_encode و json_decode استفاده نمایید.
- در صورت نیاز به unserialize آن را به کلاس های مورد نیاز محدود نمایید.
- نمونه کد آسیب پذیر:

```
<?php
    class cacheClass { function __destruct() { @unlink($this->file); } }
    $data=unserialize($_GET[“data”]);
?>
```

OWASP
The Open Web Application Security Project

Command Execution Vulnerability

آسیب پذیری اجرای دستورات خط فرمان

- عوامل آسیب پذیری: استفاده نادرست و عدم فیلتر ورودی های کاربر در توابع متصل به شل سیستم
- توابع آسیب پذیر: exec, shell_exec, system, passthru, popen, proc_open, pcntl_exec
- اجرای دستورات خط فرمان با در نظر گرفتن محدودیت های دسترسی
- درج، ویرایش و حذف فایل یا استفاده از منابع سرور برای حملات توزیع شده
- دسترسی به فایل های پیکربندی سیستم
- اجرای اکسپلویت های محلی و افزایش سطح دسترسی به سیستم
- کلید واژه ها: RCE, Remote Command Execution, Command Injection

How to prevent Command Execution vulnerability?

چگونه از آسیب پذیری اجرای دستورات خط فرمان جلوگیری کنیم؟

- از توابع اجرای دستورات خط فرمان استفاده نکنید.
- توابع حساس را توسط disable_functions محدود کنید.
- در صورت نیاز به استفاده از توابع متصل به خط فرمان، ورودی کاربر را فیلتر نمایید.
- کاراکترهای ویژه و به خصوص ` $ & * () | ; : <> را در ورودی کاربر فیلتر کنید.
- پردازش های PHP را در محیط CGI اجرا و سطح کاربری آن را محدود نمایید.
- نمونه کد آسیب پذیر:

```
<?php system(“ping -c 3 ” . $_GET[“host”]); ?>
```

OWASP
The Open Web Application Security Project

Cross-site Scripting Vulnerability

آسیب پذیری تزریق کد به مرورگر کاربران

- عوامل آسیب پذیری: عدم بررسی و فیلتر ورودی کاربر و نمایش مستقیم آن در خروجی
- توابع آسیب پذیر: امکان بروز آسیب پذیری در تمام دستورات و توابع نمایش داده وجود دارد. برای مثال: echo
- آسیب پذیری در سمت کلاینت تاثیر گذاشته و بازدید کنندگان را تهدید می نماید.
- دسترسی به اطلاعات cookie و سرقت session
- اجرای کدهای مخرب در مرورگر و سیستم بازدید کننده
- نفوذگر می‌تواند بازدید کننده را به صفحه ای دیگر منتقل نموده یا اطلاعات جعلی به او نمایش دهد.
- کلید واژه ها: XSS, Reflected XSS, Persistent XSS

OWASP
The Open Web Application Security Project

How to prevent Cross-site Scripting vulnerability?

چگونه از آسیب پذیری XSS جلوگیری کنیم؟

- در صورت نیاز به زبان نشانه گذاری بجای HTML از BBCode استفاده کنید.
- تگ های HTML را با تابع strip_tags از ورودی کاربر حذف نمایید.
- برای نمایش خروجی با استفاده از توابع htmlentities و htmlspecialchars کاراکترهای ویژه را کد کنید.
- برای ذخیره داده ها در دیتابیس از الگوریتم های encoding مانند base64 استفاده نمایید.
- ورودی کاربر را با استفاده از preg_replace بررسی و فیلتر کنید.
- نمونه کد آسیب پذیر:

```
<?php echo “Search results for: ” . $_GET[“search”]; ?>
```

OWASP
The Open Web Application Security Project

Cross-site Request Forgery Vulnerability

آسیب پذیری درخواست جعلی از کاربر تایید شده

- عوامل آسیب پذیری: اجرای فرآیند های حساس بدون بررسی و اعتبار سنجی
- توابع آسیب پذیر: محدود به تابع خاصی نبوده و ممکن است در هر شکلی از اجرای درخواست رخ دهد.
- آسیب پذیری از روش های مهندسی اجتماعی یا استفاده از آسیب پذیری **XSS** اکسپلویت می‌شود.
- صدور درخواست جعلی از کاربری که در وبسایت ورود نموده و احراز هویت شده است.
- اجرای فرآیند های تعریف شده بر اساس سطح دسترسی کاربری که اعتبار سنجی شده است.
- پرداخت صورت حساب، انتقال وجه، ایجاد یا حذف حساب کاربری، ارسال نامه، درج، ویرایش و حذف فایل
- کلید واژه ها: **CSRF, XSRF, Session riding, Sea-surf, One-click attack**

How to prevent Cross-site Request Forgery vulnerability?
چگونه از آسیب پذیری CSRF جلوگیری کنیم؟

- فرآیند های حساس را از طریق متد GET اجرا نکنید.
- جهت اعتبار سنجی درخواست کاربر، از Token استفاده نمایید.
- ارجاع دهنده درخواست HTTP_REFERER را بررسی و تصدیق نمایید.
- آسیب پذیری های XSS وبسایت را شناسایی و رفع نمایید.
- افزونه های ضد جعل درخواست را روی مرورگر کاربران با سطح دسترسی بالا فعال کنید.
- نمونه کد آسیب پذیر:

```
<?php if(is_login() && $_GET[“action”]==”newadmin”) newadmin(); ?>
```

OWASP
The Open Web Application Security Project

Unrestricted File Upload Vulnerability

آسیب پذیری بارگذاری فایل بدون محدودیت

- عوامل آسیب پذیری: عدم بررسی یا ضعف محدودیت ها برای بارگذاری فایل
- توابع آسیب پذیر: محدود به تابع خاصی نبوده و ممکن است در هر شکلی از بارگذاری فایل رخ دهد.
- آسیب پذیری بسیار خطرناک بوده و امکان هر حمله ای را به نفوذگر می‌دهد.
- بارگذاری و اجرای کدهای مخرب در سرور میزبان وب
- دسترسی به منابع سیستم و استفاده از آن برای فعالیت های غیرمجاز و حملات توزیع شده
- ایجاد Backdoor و دسترسی پایدار به سیستم
- کلید واژه ها: UFU Exploit, Arbitrary File Upload

How to prevent Unrestricted File Upload vulnerability?
چگونه از آسیب پذیری بارگذاری فایل جلوگیری کنیم؟

- مسیری که فایل ها در آن ذخیره می شوند را توسط پارامترهای مانت noexec نمایید.
- در تنظیمات وب سرور پسوند فایل های اجرایی را محدود کنید.
- در صورت بارگذاری عکس از معتبر بودن آن با استفاده از تابع getimagesize اطمینان حاصل نمایید.
- پسوند، نوع و محتوای فایل بارگذاری شده را بررسی کنید.
- نمونه کد آسیب پذیر:

```
<?php
    if(isset($_POST[“submit”]))
      move_uploaded_file($_FILES[“image”][“tmp_name”], $_FILES[“image”][“name”]);
?>
```

OWASP
The Open Web Application Security Project

Conclusion and tips for safe coding in PHP language

نتیجه گیری و نکاتی در مورد برنامه نویسی امن به زبان PHP

- به یاد داشته باشید یک کامپیوتر امن یک کامپیوتر خاموش است!
- برنامه نویسی به زبان PHP با توجه به فراوانی توابع و کلاس های آن نیاز به دقت بیشتری دارد.
- وجود آسیب پذیری ها ناشی از عدم بررسی و فیلتر داده های ورودی می باشد.
- هکرها یک قدم جلوتر از شما حرکت می‌کنند!
- برای تست نرم افزار با دید یک نفوذگر به سیستم نگاه کنید نه یک توسعه دهنده!
- پیش از عملیاتی نمودن نرم افزار، آن را با داده های واقعی تست کنید.
- از افزونه های امنیتی برای افزایش امنیت وبسایت استفاده نمایید.
- برای بررسی و رفع آسیب پذیری های نرم افزار خود، از یک متخصص امنیت کمک بگیرید.

OWASP
The Open Web Application Security Project

Thanks for your attention.

از توجه شما سپاسگزارم.

www.owasp.org/index.php/User:Saeid_Bostandoust