SPACE OF PHP SECURITY
FIRST LEVEL 101

WHOAMI

Mohammed Alharbi AKA **HitmanAlharbi**
Web developer and **security researcher**
Follow me on twitter **@HitmanF15**

WHAT IS PHP

PHP is a popular general-purpose scripting language that is especially suited to web development, it's **server-side** and **dynamically typed language**

WHAT IS PHP

- **Supports full object-oriented** programming interface
- It’s **dynamically typed** language
- **Doesn't permit** programmers to **manage memory**
- **Supports custom serialization**

PHP INPUTS

READ MORE ABUT PHP

Please check the **official manual** to **know more**
https://www.php.net/manual/en

WHY PHP SECURITY

Web development in a **pure PHP language** can be more dangerous especially if the developer **doesn’t follow a specific rules** or **doesn’t RTFM**

MY GOLDEN TIP

READ THE FANTASTIC MANUAL (**RTFM**)

RTFM

eval

(PHP 4, PHP 5, PHP 7, PHP 8)
eval — Evaluate a string as PHP code

Description

```
eval(string $code): mixed
```

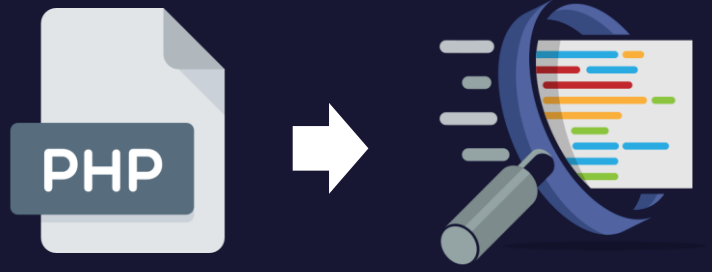
Evaluates the given **code** as PHP.

Caution The **eval()** language construct is *very dangerous* because it allows execution of arbitrary PHP code. *Its use thus is discouraged.* If you have carefully verified that there is no other option than to use this construct, pay special attention *not to pass any user provided data* into it without properly validating it beforehand.

STATIC CODE ANALYSIS

Static code analysis (Also known as **whitebox testing**) is a method of debugging that is done by **examining the code without executing** the program

STATIC CODE ANALYSIS



VULNERABILITES
UNSAFE USE
BUGS

UNSAFE USE

PHP is secure, but the **problem happened** when the developer **use** some of **PHP functions** in **unsafe way,** or write **unsafe code**

UNSAFE USE EXAMPLE

Print function used to **print a text** in the page, but unsafe use of this function can result a **XSS vulnerability**

XSS

Cross-Site Scripting (XSS) is a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites

XSS EXAMPLE

```
echo( $_GET["name"] );
```

UNSAFE USE EXAMPLE

Include function used to **add a file** in the workspace, but unsafe use of this function can result an **FI vulnerability**

File Inclusion (FI) vulnerability allows an attacker to include local or remote files and sometimes can **execute them** in the workspace

FI EXAMPLE

```
include( $_GET["file"] );
```

UNSAFE USE EXAMPLE

Curl function used to **send HTTP requests**, but unsafe use of this function can result a **SSRF vulnerability**

SSRF

Server-side request forgery (SSRF) is an attack that allows attackers to **send malicious requests** to other systems **via a vulnerable web server**

SSRF EXAMPLE

```
file_get_contents( $_GET["url"] );
```

UNSAFE USE EXAMPLE

Unsafe use of **Strict and Loose Comparisons** can result a **Type Juggling vulnerability**

TYPE JUGGLING

Type Juggling is a vulnerability wherein an object is accessed as the incorrect type, allowing an attacker to potentially **bypass authentication** or **undermine the type safety of an application**

TYPE JUGGLING EXAMPLE

```
if( $_GET["pass"] == md5("password")){..}
```

UNSAFE USE EXAMPLE

PLEASE RTFM FOR MORE ...

CHALLENGE TIME

IT THIS CODE SAFE

Windows environment

```
system("echo hey " . $_GET["name"] );
```

WHAT ABOUT NOW

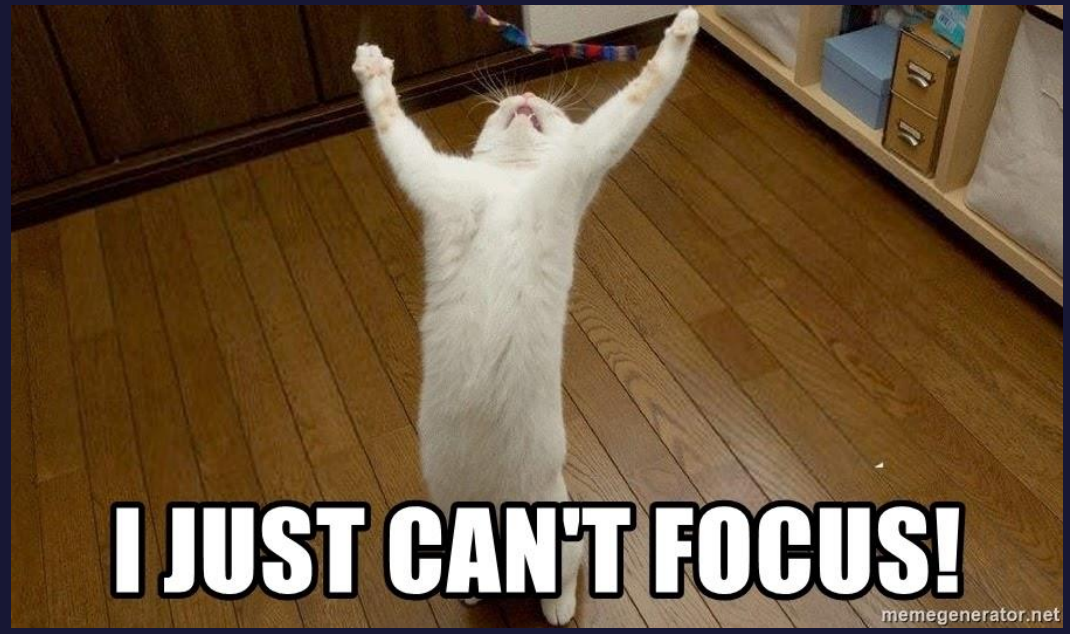
```
$name = str_replace('|', '', $_GET["name"]);

system('echo hey ' . $ name);
```

WHAT ABOUT NOW

```
$name = str_replace('|', '', $_GET["name"]);
$name = str_replace('>', '', $_GET["name"]);
system('echo hey ' . $ name);
```

FOCUS



WHAT ABOUT NOW

```
$name = str_replace('|', '', $_GET["name"]);
$name = str_replace('>', '', $name);
system('echo hey ' . $ name);
```


WARNING

FIX

Why are we using **system function** to **print a text!!** we can use **print** or **echo** directly

IT THIS CODE SAFE

```
echo("Hey " . $_GET["name"] );
```

WHAT ABOUT NOW

```
$name = htmlspecialchars($_GET["name"]);
echo("Hey " . $name);
```

NOTE

It’s good to use **htmlspecialchars** to avoid **XSS vulnerabilities** but this solution **is not enough,** we should do some **sanitization**

IT THIS CODE SAFE

```
echo("<a href='" . $_GET["url"] . "'>Visit</a>");
```

WHAT ABOUT NOW

```
$url = htmlspecialchars($_GET["url"]);
echo("<a href='" . $url . "'>Visit</a>");
```


WARNING
WARNING
WARNING
WARNING
WARNING
WARNING
WARNING
WARNING
WARNING
WARNING
WARNING
WARNING

FIX

QUESTIONS TIME