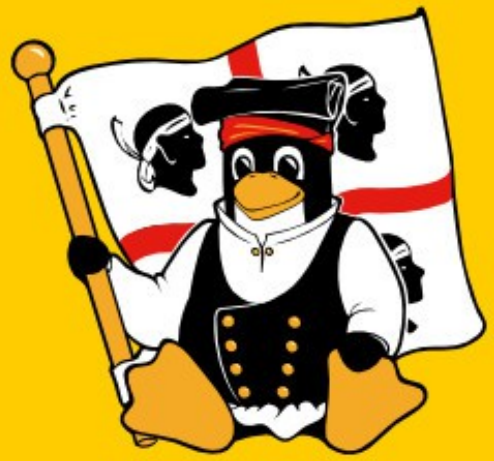
PHP Security

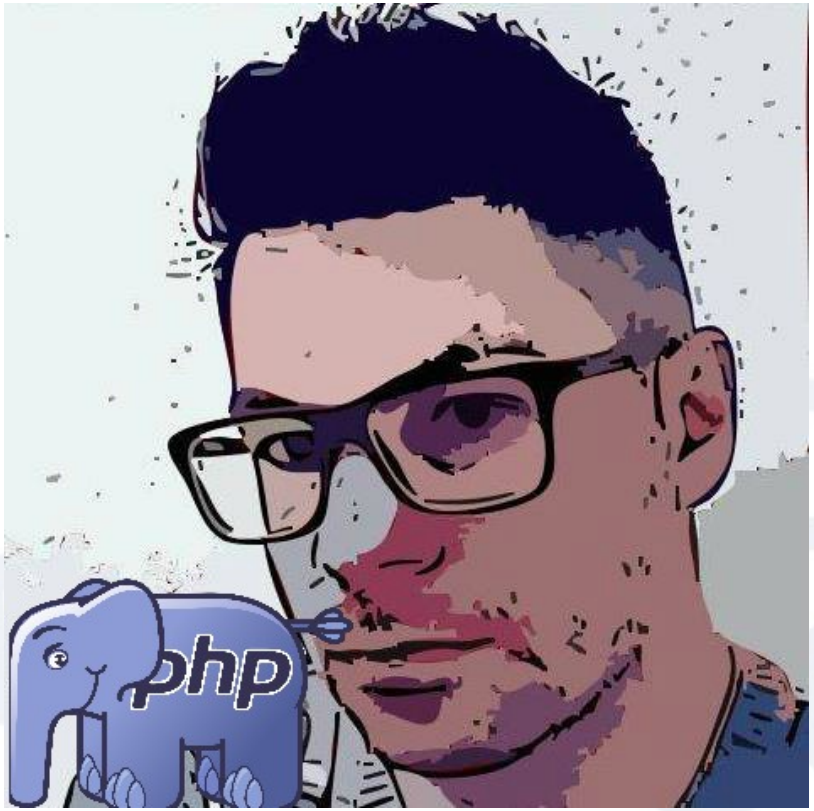
di Dario Fadda



GULCh

Gruppo Utenti Linux Cagliari h...?

Chi Sono?

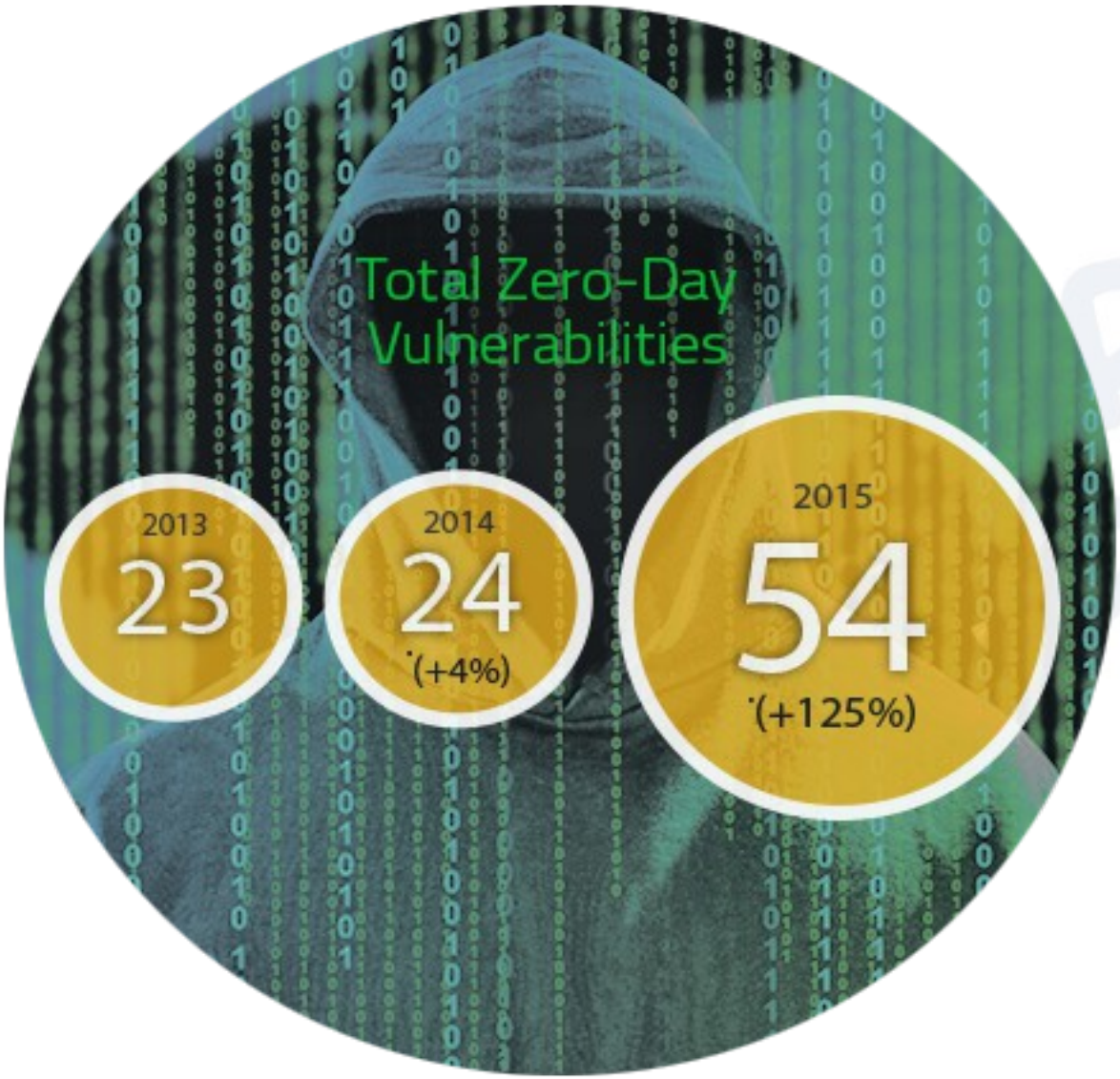


- Dario Fadda
- Dal 2005 Gulch e Linux Day
- 8 anni con PHP
- Scrivo su spcnet.it e dariofadda.it

Di cosa parliamo?

1) Qualche dato
2) Come proteggersi
3) Le Passwords
4) PHP e web server
5) Valida sempre tutto
6) SQL Injection
7) Usa PDO
8) Offusca i dati
9) Nascondi PHP

1) Qualche dato statistico



Vulnerabilità Zero-Day

Vulnerabilità in ¾ dei siti Web

100 Milioni di siti fake scoperti nel 2015

82% dei siti usa PHP,
Ottobre 2016 – w3Techs.com

PHP:

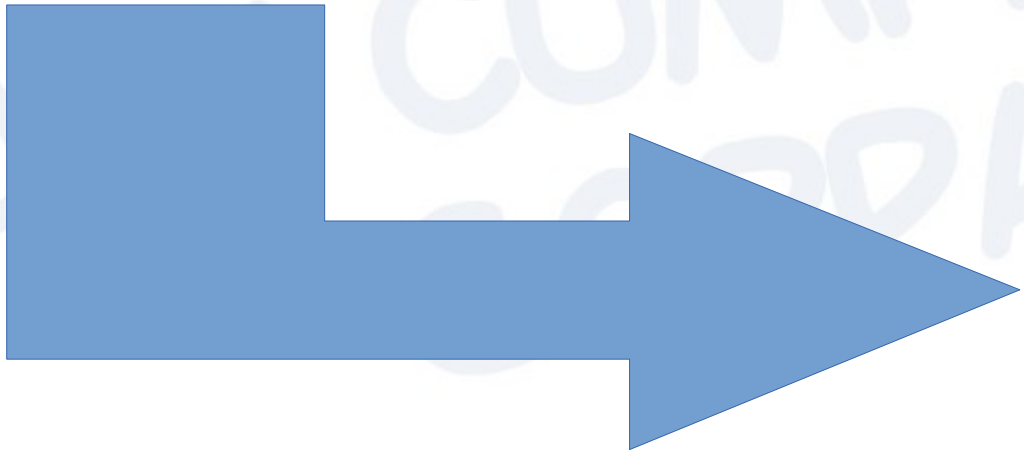
- **v. 5 97,4%**
- **v. 7 1,5%**
- **v. 4 1,1%**
- **v. 3 0,1%**

2) Come proteggersi?

Controllo degli accessi: con password

Crittografia per la conservazione dei dati

Log e monitor di ogni singola azione



Evitare codice malevolo

Vulnerabilità già note (*di codice!*) sono alla base della maggior parte dei siti web rilevati come VULNERABILI

3) Le Passwords

INSICURO

sha1();

md5();

SICURO

bcrypt

```
password_hash('bella_password'
, PASSWORD_DEFAULT);
// Stringa di 60-caratteri
```

4) PHP e web server

L'abitudine quotidiana:

&

La BUONA abitudine:

```
$:/# sudo apt-get install
       apache2-mpm-event
       libapache2-mod-fastcgi
       php5-fpm
$:/# sudo a2enmod actions alias
       fastcgi
```

```
<VirtualHost *:80>
    Action php5-fcgi /php5-fcgi
    Alias /php5-fcgi /usr/lib/cgi-bin/php5-fcgi
    FastCgiExternalServer /usr/lib/cgi-bin/php5-
fcgi -socket /var/run/php5-fpm.sock -idle-timeout
120 -pass-header Authorization
    <FilesMatch "\.php$">
        SetHandler  php5-fcgi
    </FilesMatch>
</VirtualHost>
```

Facciamo girare tutte le richieste PHP verso il processo **PHP-FPM** per migliorare il severo utilizzo della memoria di **mod_php**

5) Valida sempre tutto

```php
if (!preg_match("/^[0-9]{1,2}$/", $_GET['mese'])){
    // errore errore!!
}
if (!preg_match("/^[0-9]{1,2}$/", $_GET['giorno'])){
    // errore errore!!
}
if (!preg_match("/^[0-9]{4}$/", $_GET['anno'])){
    // errore errore!!
}


// purifichiamo l'HTML
htmlentities();

// validiamo le email
filter_var('sgamgee@example.com', FILTER_VALIDATE_EMAIL);
```

6) SQL Injection

Purtroppo di uso comune:

```
<?php
$user = $_POST['user'];
$pass = $_POST['password'];
$sql = mysqli_query("
          SELECT * FROM admins WHERE
          username = '$user' AND
          password = '$pass'
       ");
?>
```

Filtrare sempre i dati:

```
mysqli_real_escape_string();

settype($id, 'integer');
```

Attacco:

```
$user = ' or '1'='1'; DELETE … UPDATE …
```

7) Usa PDO

```php
<?php
// PDO + MySQL
$pdo = new PDO('
       mysql:host=example.com;dbname=database',
       'utente', 'password'
       );
$id = $_GET['id'];

$statement = $pdo->prepare("
       SELECT * FROM tbl_utenti WHERE id = ?"
       );

$statement->bind_param("i", $id);
$statement->execute();
```

i = INTEGER
s = STRING
d = DOUBLE
b = BLOB

8) Offuscare i dati

Alcuni metodi sempre utili:

- Utilizzare nomi di variabili difficili da scoprire;
- Non utilizzare i nomi delle variabili nelle **URLs** o nei **forms**;
- **BCRYPT** per le informazioni sensibili (password, carte c., ...);
- Utilizzare messaggi di errore generici;

9) Nascondi PHP

Modifichiamo il .htaccess:

- #1. Confondi i file .php con altri linguaggi

```
AddType application/x-httpd-php .asp .py .pl
```

- #2. Confondi i file .php con nuove estensioni

```
AddType application/x-httpd-php .bop .foo .133t
```

- #3. Confondi i file .php con pagine HTML

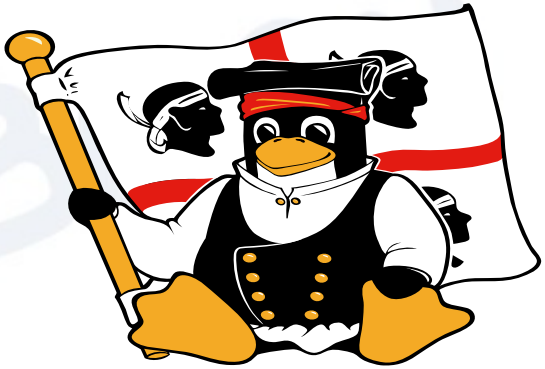
```
AddType application/x-httpd-php .htm .html
```

http://php.net/manual/en/security.hiding.php

GRAZIE A TUTTI!

25.08.2016

Buon 25esimo compleanno
GNU/Linux



Buon 20esimo compleanno
GOPPAI

Grazie prof. Giulio Concas...