By Falko Timme
Published: 2008-06-22 18:17

Intrusion Detection For PHP Applications With PHPIDS

Version 1.0
Author: Falko Timme <ft [at] falkotimme [dot] com>
Last edited 06/04/2008

This tutorial explains how to set up **PHPIDS** on a web server with Apache2 and PHP5. PHPIDS (PHP-Intrusion Detection System) is a simple to use, well structured, fast and state-of-the-art security layer for your PHP based web application. The IDS neither strips, sanitizes nor filters any malicious input, it simply recognizes when an attacker tries to break your site and reacts in exactly the way you want it to. Based on a set of approved and heavily tested filter rules any attack is given a numerical impact rating which makes it easy to decide what kind of action should follow the hacking attempt. This could range from simple logging to sending out an emergency mail to the development team, displaying a warning message for the attacker or even ending the user(TM)s session.

I do not issue any guarantee that this will work for you!

1 Preliminary Note

I have tested this on a Debian Etch LAMP system with Apache2 and PHP5 and the IP address *192.168.0.100*. The Apache user and group on Debian Etch is *www-data*, so if you are on a different distribution, the Apache user and group might be different. The location of *php.ini* (*/etc/php5/apache2/php.ini* on Debian Etch) might differ as well.

I'm using a virtual host with the document root */var/www/web1/web* in this example.

2 Installing PHPIDS

For security reasons, I want to install PHPIDS outside of the document root, so I create the directory */var/www/web1/phpids*:

```
mkdir /var/www/web1/phpids
```

Then I install PHPIDS as follows (at the time of this writing the latest version was 0.4.7) - of all the contents of the *phpids-0.4.7.tar.gz* file, we only need the *lib/* directory:

```
cd /tmp

wget http://php-ids.org/files/phpids-0.4.7.tar.gz

tar xvfz phpids-0.4.7.tar.gz

cd phpids-0.4.7

mv lib/ /var/www/web1/phpids/
```

Now I change to the directory */var/www/web1/phpids/lib/IDS*...

```
cd /var/www/web1/phpids/lib/IDS
```

... and make the *tmp/* directory (which will hold the PHPIDS log file) writable for the Apache user and group:

```
chown -R www-data:www-data tmp/
```

Next we configure the PHPIDS configuration file (*Config.ini*):

```
cd Config/

vi Config.ini
```

I'm using the default configuration here, all I did was to adjust the paths:

```
; PHPIDS Config.ini

; General configuration settings

; !!!DO NOT PLACE THIS FILE INSIDE THE WEB-ROOT IF DATABASE CONNECTION DATA WAS ADDED!!!

[General]

    filter_type     = xml
    filter_path     = /var/www/web1/phpids/lib/IDS/default_filter.xml
    tmp_path        = /var/www/web1/phpids/lib/IDS/tmp
    scan_keys       = false

    exceptions[]    = __utmz
    exceptions[]    = __utmc

; If you use the PHPIDS logger you can define specific configuration here

[Logging]

    ; file logging
    path            = /var/www/web1/phpids/lib/IDS/tmp/phpids_log.txt

    ; email logging

    ; note that enabling safemode you can prevent spam attempts,
    ; see documentation
    recipients[]    = test@test.com.invalid
    subject         = "PHPIDS detected an intrusion attempt!"
    header                  = "From: <PHPIDS> info@php-ids.org"
    safemode        = true
    allowed_rate    = 15
```

```
    ; database logging

    wrapper        = "mysql:host=localhost;port=3306;dbname=phpids"
    user           = phpids_user
    password       = 123456
    table          = intrusions

; If you would like to use other methods than file caching you can configure them here

[Caching]

    ; caching:     session|file|database|memcached|none
    caching        = file
    expiration_time = 600

    ; file cache
    path           = /var/www/web1/phpids/lib/IDS/tmp/default_filter.cache

    ; database cache
    wrapper        = "mysql:host=localhost;port=3306;dbname=phpids"
    user           = phpids_user
    password       = 123456
    table          = cache

    ; memcached
    ;host          = localhost
    ;port          = 11211
    ;key_prefix    = PHPIDS
    ;tmp_path      = /var/www/web1/phpids/lib/IDS/tmp/memcache.timestamp
```

3 Using PHPIDS

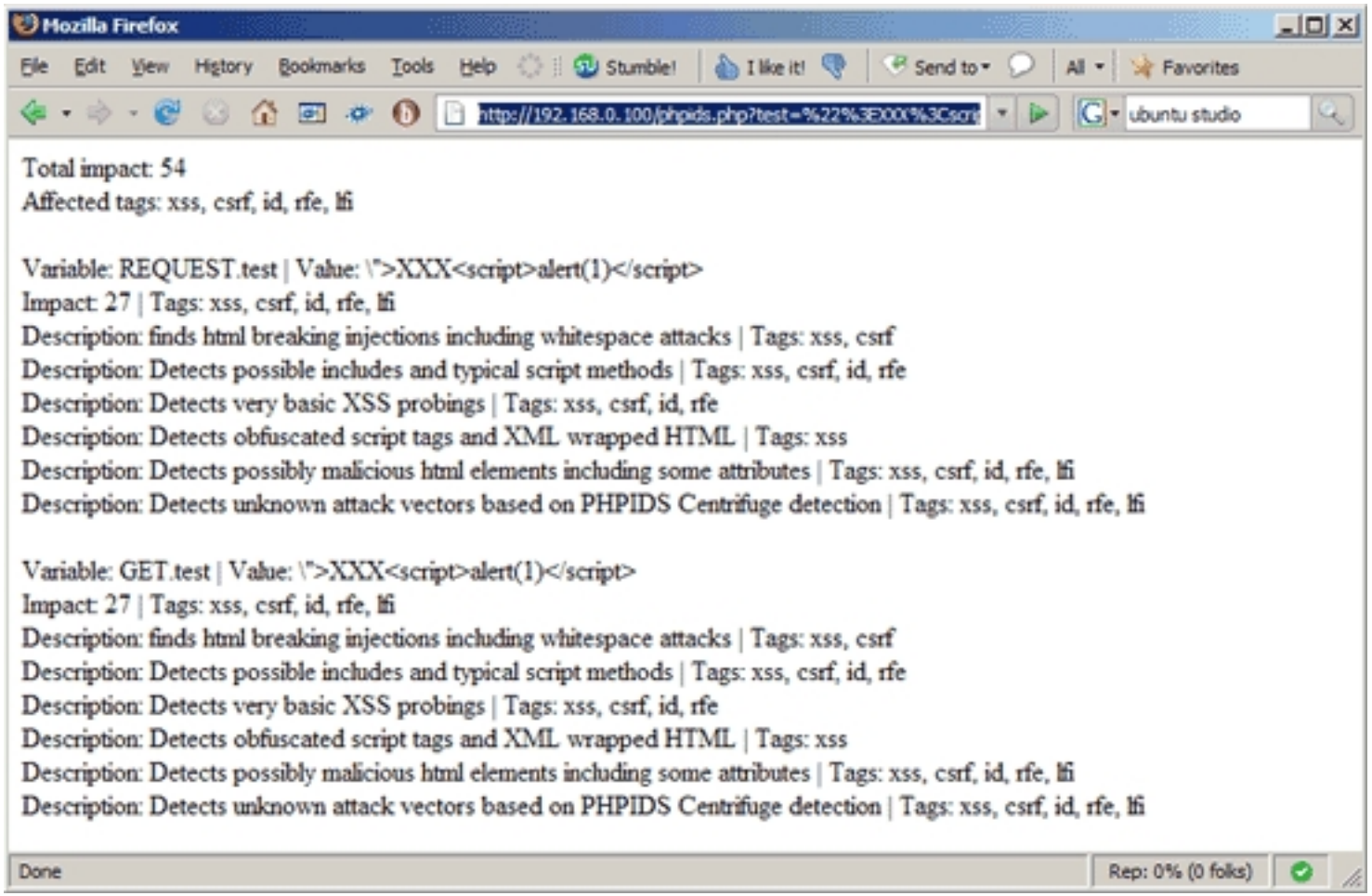
We will now create the file */var/www/web1/web/phpids.php* which will call PHPIDS for us (we will later on prepend that file to all our PHP files so that our PHP files can make use of PHPIDS automatically):

```
vi /var/www/web1/web/phpids.php
```

```
<?php
set_include_path(
  get_include_path()
  . PATH_SEPARATOR
  . '/var/www/web1/phpids/lib'
 );

 require_once 'IDS/Init.php';
 $request = array(
    'REQUEST' => $_REQUEST,
    'GET' => $_GET,
    'POST' => $_POST,
    'COOKIE' => $_COOKIE
 );
 $init = IDS_Init::init('/var/www/web1/phpids/lib/IDS/Config/Config.ini');
 $ids = new IDS_Monitor($request, $init);
 $result = $ids->run();

 if (!$result->isEmpty()) {
  // Take a look at the result object
  echo $result;
  require_once 'IDS/Log/File.php';
  require_once 'IDS/Log/Composite.php';

  $compositeLog = new IDS_Log_Composite();
  $compositeLog->addLogger(IDS_Log_File::getInstance($init));
```

```
  $compositeLog->execute($result);
 }
?>
```

Now when you call that file in a browser, (e.g. *http://192.168.0.100/phpids.php*), you will see a blank page. But if you try to append some malicious parameters to the URL (e.g. *http://192.168.0.100/phpids.php?test=%22%3EXXX%3Cscript%3Ealert(1)%3C/script%3E*), PHPIDS will detect this and print its findings in the browser:



Now we have to find a way to make our PHP scripts use PHPIDS. Of course, you don't want to modify all your PHP scripts (you could have hundreds of them...). Fortunately, there's a better way: we can tell PHP to prepend a PHP script whenever a PHP script is called. For example, if we call the script `info.php` in a browser, PHP would first execute `phpids.php` and then `info.php`, and we don't even have to modify `info.php`.

We can do this by using PHP's `auto_prepend_file` parameter. We can either set this in our `php.ini` (this is a global setting which is valid for all PHP web sites on the server), or in an `.htaccess` file (this is a setting valid only for the web site in question):**php.ini**

Open your php.ini (e.g. */etc/php5/apache2/php.ini*), and set *auto_prepend_file* to */var/www/web1/web/phpids.php*:

```
vi /etc/php5/apache2/php.ini
```

```
[...]
auto_prepend_file = /var/www/web1/web/phpids.php
[...]
```

Restart Apache afterwards:

```
/etc/init.d/apache2 restart
```

.htaccess

Instead of modifying *php.ini* (which is a global change, i.e., the change is valid for all web sites that use PHP on the server), you can instead use an *.htaccess* file (so the setting would be valid only for the web site for which you create the *.htaccess* file):

```
vi /var/www/web1/web/.htaccess
```

```
php_value auto_prepend_file /var/www/web1/web/phpids.php
```

Please make sure that the vhost for the web site in */var/www/web1/web* contains something like this (otherwise the *php_value* line in the *.htaccess* file will be ignored) (if you have to modify the vhost, please don't forget to restart Apache):

```
<Directory /var/www/web1/web/>
```

```
AllowOverride All
</Directory>
```

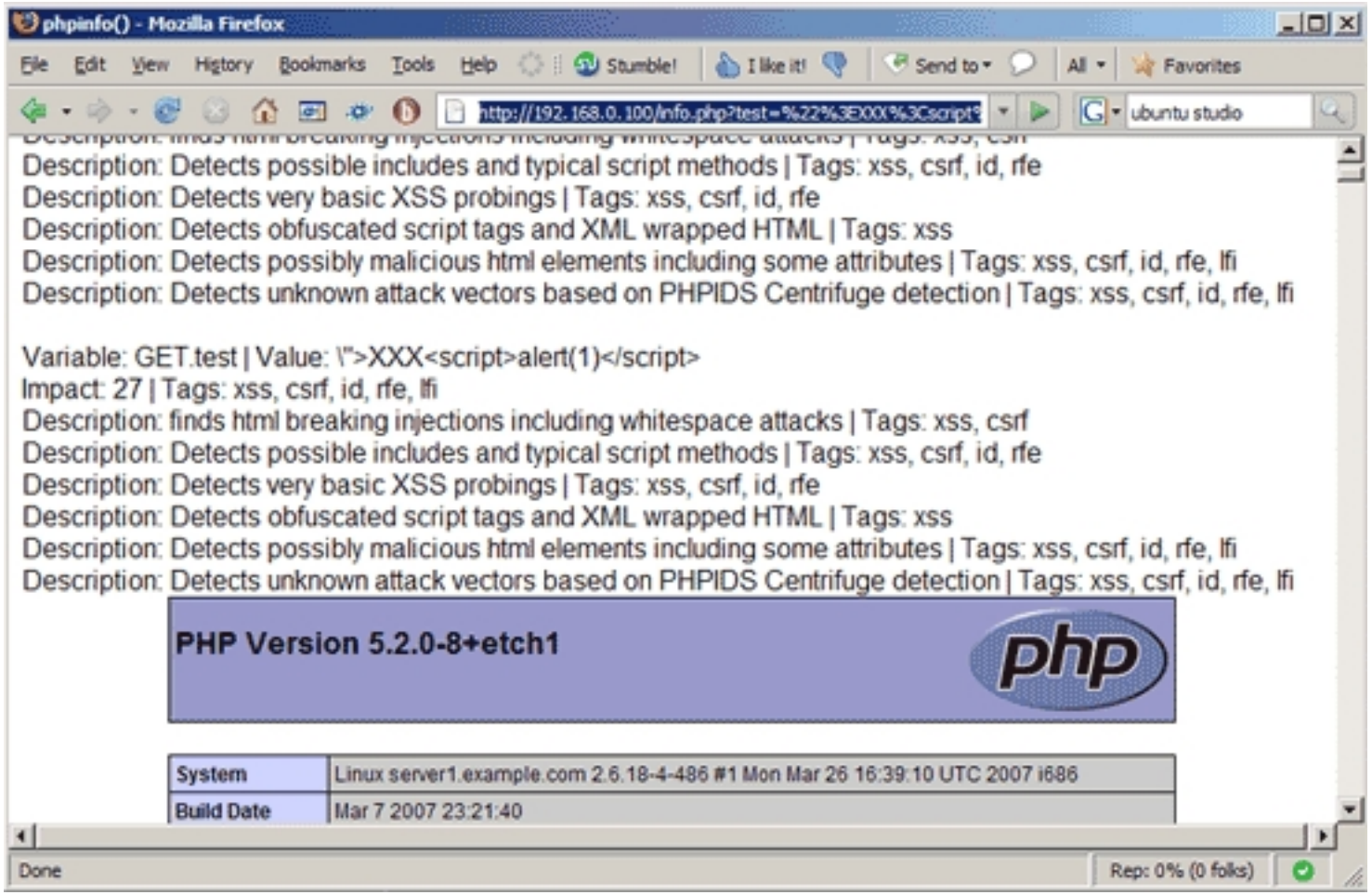
Now we create a simple PHP file, */var/www/web1/web/info.php*:

```
vi /var/www/web1/web/info.php
```

```
<?php
phpinfo();
?>
```

Call that file in a browser (*http://192.168.0.100/info.php*), and you should see the normal *phpinfo()* output.

Now append some malicious parameters to the URL (e.g. *http://192.168.0.100/info.php?test=%22%3EXXX%3Cscript%3Ealert(1)%3C/script%3E*), and you should find a PHPIDS report before the *phpinfo()* output (because */var/www/web1/web/phpids.php* was executed before */var/www/web1/web/info.php*):



PHPIDS logs to */var/www/web1/phpids/lib/IDS/tmp/phpids_log.txt*, so you should see something in the log now:

```
cat /var/www/web1/phpids/lib/IDS/tmp/phpids_log.txt
```

```
"192.168.0.200",2008-06-04T17:36:08+02:00,54,"xss csrf id rfe
```

*lfi","REQUEST.test=%5C%22%3EXXX%3Cscript%3Ealert%281%29%3C%2Fscript%3E
GET.test=%5C%22%3EXXX%3Cscript%3Ealert%281%29%3C%2Fscript%3E",
"%2Finfo.php%3Ftest%3D%2522%253EXXX%253Cscript%253Ealert%281%29%253C%2Fscript%253E"*

Now by observing that log you learn what hackers are trying to do to your PHP applications, and you can try to harden your applications.

To add another level of security, we can stop our PHP scripts from executing if PHPIDS find that they are under attack: we simply add something like *die('<h1>Go away!</h1>');* to the *if (!$result->isEmpty()) {}* section of the */var/www/web1/web/phpids.php* script:

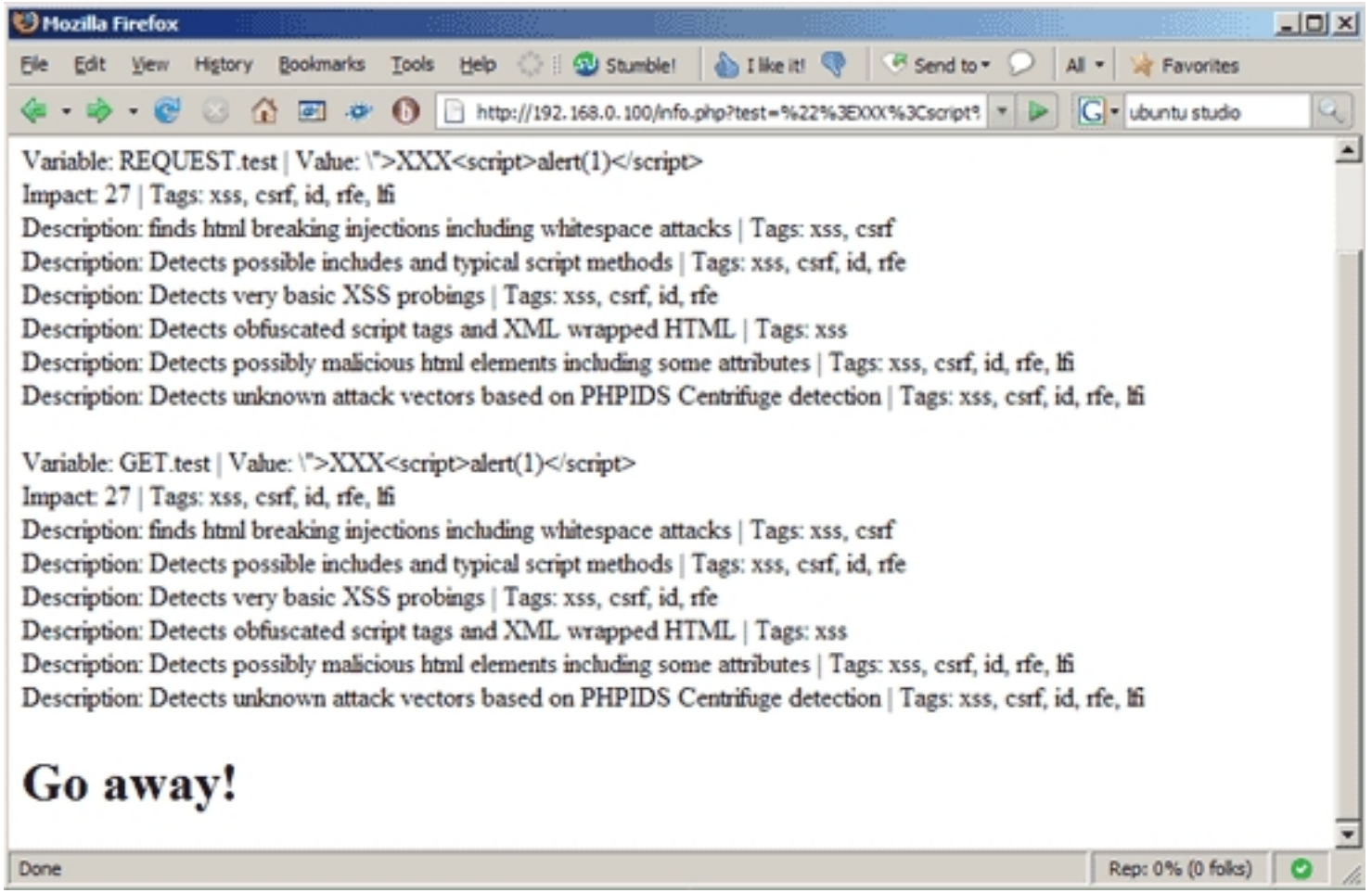
```
vi /var/www/web1/web/phpids.php
```

```
<?php
set_include_path(
   get_include_path()
   . PATH_SEPARATOR
   . '/var/www/web1/phpids/lib'
  );

  require_once 'IDS/Init.php';
  $request = array(
      'REQUEST' => $_REQUEST,
      'GET' => $_GET,
      'POST' => $_POST,
      'COOKIE' => $_COOKIE
  );
  $init = IDS_Init::init('/var/www/web1/phpids/lib/IDS/Config/Config.ini');
  $ids = new IDS_Monitor($request, $init);
  $result = $ids->run();

  if (!$result->isEmpty()) {
   // Take a look at the result object
```



```
  echo $result;
  require_once 'IDS/Log/File.php';
  require_once 'IDS/Log/Composite.php';

  $compositeLog = new IDS_Log_Composite();
  $compositeLog->addLogger(IDS_Log_File::getInstance($init));
  $compositeLog->execute($result);

  die('<h1>Go away!</h1>');
 }
?>
```

If there's no attack, the scripts are executed, but if PHPIDS finds an attack, it prevents the scripts from being executed and displays a message to the hackers:



4 Links

- PHPIDS: **http://php-ids.org**
- PHP: **http://www.php.net**
- Apache: **http://httpd.apache.org**