PHP插件获取网页返回的html源码

Lou00 / 2019-08-17 10:23:00 / 浏览数 4348 安全技术 WEB安全 顶(0) 踩(0)

起因

想通过php扩展获取到页面返回的response

ob_start的源码实现

先看看ob_start的实现
在main/output.c和main/php_output.h下

```
PHP_FUNCTION(ob_start)
{
   zval *output_handler = NULL;
   zend_long chunk_size = 0;
   zend_long flags = PHP_OUTPUT_HANDLER_STDFLAGS;

   if (zend_parse_parameters(ZEND_NUM_ARGS(), "|z/ll", &output_handler, &chunk_size, &flags) == FAILURE) {
       return;
   }

   if (chunk_size < 0) {
       chunk_size = 0;
   }

   if (php_output_start_user(output_handler, chunk_size, flags) == FAILURE) {
       php_error_docref("ref.outcontrol", E_NOTICE, "failed to create buffer");
       RETURN_FALSE;
   }
   RETURN_TRUE;
}
```

跟进到最后会发现php_output_handler_start函数

```
PHPAPI int php_output_handler_start(php_output_handler *handler)
{
   ...
   /* zend_stack_push returns stack level */
   handler->level = zend_stack_push(&OG(handlers), &handler);
   OG(active) = handler;
   return SUCCESS;
}
```

其中OG是一个叫output_globals的全局变量ob_start()后的缓冲区
经过调试,发现所有于输出有关的函数都会调用一个叫php_output_op的函数

```
static inline void php_output_op(int op, const char *str, size_t len)
{
   php_output_context context;
   ...
   if (OG(active) && (obh_cnt = zend_stack_count(&OG(handlers)))) {
       context.in.data = (char *) str;
       context.in.used = len;
       ...
   } else {
       context.out.data = (char *) str;
       context.out.used = len;
   }
   ...
}
```

通过判断OG(active)是否为NULL来决定进不进入缓存区
接着来看看字符串如何进入缓冲区

```
static inline int php_output_handler_append(php_output_handler *handler, const php_output_buffer *buf)
{
   if (buf->used) {
       OG(flags) |= PHP_OUTPUT_WRITTEN;
       /* store it away */
       //■■■■■■■■■
       if ((handler->buffer.size - handler->buffer.used) <= buf->used) {
           size_t grow_int = PHP_OUTPUT_HANDLER_INITBUF_SIZE(handler->size);
           size_t grow_buf = PHP_OUTPUT_HANDLER_INITBUF_SIZE(buf->used - (handler->buffer.size - handler->buffer.used));
           size_t grow_max = MAX(grow_int, grow_buf);

           handler->buffer.data = erealloc(handler->buffer.data, handler->buffer.size + grow_max);
           handler->buffer.size += grow_max;
       }
       //■■■■■■■
       memcpy(handler->buffer.data + handler->buffer.used, buf->data, buf->used);
       handler->buffer.used += buf->used;

       /* chunked buffering */
       if (handler->size && (handler->buffer.used >= handler->size)) {
           /* store away errors and/or any intermediate output */
           return OG(running) ? 1 : 0;
       }
   }
   return 1;
}
```

编写插件

思路

可以和output_globals.active->buffer相似,创造一个全局的缓存区
在MINIT阶段初始化这个全局变量并hook各输出函数的opcode,写入缓冲区
在RSHUTDOWN阶段将全局变量的数据保存在文件内

创建一个插件

在php的源码下进入ext目录,输入

```
./ext_skel --extname=myext
```

全局变量的定义与初始化

编辑php_hook_output_ext.h
先来看一下output_globals.active->buffer的结构

```
typedef struct _php_output_buffer {
   char *data;
   size_t size;
   size_t used;
   uint free:1;
   uint _reserved:31;
} php_output_buffer;
```

在上文的php_output_handler_append函数中可看到只用了前3个
于是编写全局变量如下

```
ZEND_BEGIN_MODULE_GLOBALS(myext)
   char *data; //■■■
   size_t size; //■■■■■
   size_t used; //■■■■
ZEND_END_MODULE_GLOBALS(myext)
```

完成定义,在hook_output_ext.c下进行初始化与析构

```
static void php_myext_globals_ctor(zend_myext_globals *G TSRMLS_DC)
{
   G->data = NULL;
   G->size = 0;
   G->used = 0;
}
```

```
static void php_myext_globals_dtor(zend_myext_globals *G TSRMLS_DC)
{
   efree(G->data);
}
```

并分别在MINIT阶段和RSHUTDOWN调用

hook opcode

这里以ZEND_ECHO这条opcode为例
当php执行echo xxxx;时会调用这条opcode
关于如何hook具体可以参考https://xz.aliyun.com/t/4214#toc-2
这里主要讲hook后数据的处理

```
static int get_data(char *str, size_t str_len)
{
   if(str_len){
       //size■■■■■■■■■■■■
       if ((MYEXT_G(size) - MYEXT_G(used)) <= str_len){
           size_t grow_int = PHP_OUTPUT_HANDLER_INITBUF_SIZE(MYEXT_G(size));
           size_t grow_buf = PHP_OUTPUT_HANDLER_INITBUF_SIZE(str_len - (MYEXT_G(size) - MYEXT_G(used)));
           size_t grow_max = MAX(grow_int, grow_buf);
           MYEXT_G(data) = erealloc(MYEXT_G(data), MYEXT_G(size) + grow_max);
           MYEXT_G(size) += grow_max;
       }
       memcpy(MYEXT_G(data) + MYEXT_G(used), str, str_len);
       MYEXT_G(used) += str_len;
   }
   return 1;
}

static int hook_echo(ZEND_OPCODE_HANDLER_ARGS)
{
   zend_op *opline = execute_data->opline;
   zval *z = EX_CONSTANT(opline->op1);
   if (Z_TYPE_P(z) == IS_STRING) {
       zend_string *str = Z_STR_P(z);
       if (ZSTR_LEN(str) != 0) {
           get_data(ZSTR_VAL(str), ZSTR_LEN(str));
       }
   } else {
       zend_string *str = _zval_get_string_func(z);

       if (ZSTR_LEN(str) != 0) {
           get_data(ZSTR_VAL(str), ZSTR_LEN(str));
       }
       zend_string_release(str);
   }
   return ZEND_USER_OPCODE_DISPATCH;
}
```

可以看到get_data是直接根据php_output_handler_append改的
hookecho是根据ZEND_ECHO的一个hander编成的

```
static ZEND_OPCODE_HANDLER_RET ZEND_FASTCALL ZEND_ECHO_SPEC_CV_HANDLER(ZEND_OPCODE_HANDLER_ARGS)
{
   USE_OPLINE

   zval *z;

   SAVE_OPLINE();
   z = _get_zval_ptr_cv_undef(execute_data, opline->op1.var);

   if (Z_TYPE_P(z) == IS_STRING) {
       zend_string *str = Z_STR_P(z);

       if (ZSTR_LEN(str) != 0) {
           zend_write(ZSTR_VAL(str), ZSTR_LEN(str));
       }
```

```
   } else {
       zend_string *str = _zval_get_string_func(z);

       if (ZSTR_LEN(str) != 0) {
           zend_write(ZSTR_VAL(str), ZSTR_LEN(str));
       } else if (IS_CV == IS_CV && UNEXPECTED(Z_TYPE_P(z) == IS_UNDEF)) {
           GET_OP1_UNDEF_CV(z, BP_VAR_R);
       }
       zend_string_release(str);
   }

   ZEND_VM_NEXT_OPCODE_CHECK_EXCEPTION();
}
```

文件的保存

在RSHUTDOWN处保存,文件名可以根据时间\,如果是用apache或者nginx起的话,默认是要将文件放在web根目录里否则要更改相关配置

代码(demo)

```
//php_myext.h
/*
 +----------------------------------------------------------------------+
 | PHP Version 7                                                        |
 +----------------------------------------------------------------------+
 | Copyright (c) 1997-2018 The PHP Group                                |
 +----------------------------------------------------------------------+
 | This source file is subject to version 3.01 of the PHP license,      |
 | that is bundled with this package in the file LICENSE, and is        |
 | available through the world-wide-web at the following url:           |
 | http://www.php.net/license/3_01.txt                                  |
 | If you did not receive a copy of the PHP license and are unable to   |
 | obtain it through the world-wide-web, please send a note to          |
 | license@php.net so we can mail you a copy immediately.               |
 +----------------------------------------------------------------------+
 | Author:      lou00                                                   |
 +----------------------------------------------------------------------+
*/

/* $Id$ */

#ifndef PHP_MYEXT_H
#define PHP_MYEXT_H

extern zend_module_entry myext_module_entry;
#define phpext_myext_ptr &myext_module_entry

#define PHP_MYEXT_VERSION "0.1.0" /* Replace with version number for your extension */

#ifdef PHP_WIN32
#   define PHP_MYEXT_API __declspec(dllexport)
#elif defined(__GNUC__) && __GNUC__ >= 4
#   define PHP_MYEXT_API __attribute__ ((visibility("default")))
#else
#   define PHP_MYEXT_API
#endif

#ifdef ZTS
#include "TSRM.h"
#endif

/*
   Declare any global variables you may need between the BEGIN
   and END macros here:

ZEND_BEGIN_MODULE_GLOBALS(myext)
   zend_long  global_value;
   char *global_string;
ZEND_END_MODULE_GLOBALS(myext)
```

```
*/

/* Always refer to the globals in your function as MYEXT_G(variable).
  You are encouraged to rename these macros something shorter, see
  examples in any other php module directory.
*/

#if defined(ZTS) && defined(COMPILE_DL_MYEXT)
ZEND_TSRMLS_CACHE_EXTERN()
#endif

#endif  /* PHP_MYEXT_H */


/*
* Local variables:
* tab-width: 4
* c-basic-offset: 4
* End:
* vim600: noet sw=4 ts=4 fdm=marker
* vim<600: noet sw=4 ts=4
*/


# define MYEXT_G(v) ZEND_MODULE_GLOBALS_ACCESSOR(myext,v)
ZEND_BEGIN_MODULE_GLOBALS(myext)
 char *data;
   size_t size;
 size_t used;
ZEND_END_MODULE_GLOBALS(myext)
# define ZEND_OPCODE_HANDLER_ARGS zend_execute_data *execute_data
PHP_FUNCTION(confirm_myext_compiled);
static int hookecho(ZEND_OPCODE_HANDLER_ARGS);
static int get_data(char *str, size_t str_len);
static void init_myext_global();

//myext.c
/*
 +----------------------------------------------------------------------+
 | PHP Version 7                                                        |
 +----------------------------------------------------------------------+
 | Copyright (c) 1997-2018 The PHP Group                                |
 +----------------------------------------------------------------------+
 | This source file is subject to version 3.01 of the PHP license,      |
 | that is bundled with this package in the file LICENSE, and is        |
 | available through the world-wide-web at the following url:           |
 | http://www.php.net/license/3_01.txt                                  |
 | If you did not receive a copy of the PHP license and are unable to   |
 | obtain it through the world-wide-web, please send a note to          |
 | license@php.net so we can mail you a copy immediately.               |
 +----------------------------------------------------------------------+
 | Author:      lou00                                                   |
 +----------------------------------------------------------------------+
*/

/* $Id$ */

#ifdef HAVE_CONFIG_H
#include "config.h"
#endif


#include "php.h"
#include "php_ini.h"
#include "ext/standard/info.h"
#include "php_myext.h"
#include "ext/standard/head.h"
#include "ext/standard/url_scanner_ex.h"
#include "main/php_output.h"
```

```
#include "SAPI.h"
#include "zend_stack.h"

static int le_myext;

//resgin from TRSM
ZEND_DECLARE_MODULE_GLOBALS(myext);

PHP_FUNCTION(confirm_myext_compiled)
{
   char *arg = NULL;
   size_t arg_len, len;
   zend_string *strg;

   if (zend_parse_parameters(ZEND_NUM_ARGS(), "s", &arg, &arg_len) == FAILURE) {
       return;
   }

   strg = strpprintf(0, "Congratulations! You have successfully modified ext/%.78s/config.m4. Module %.78s is now compiled int

   RETURN_STR(strg);
}

static void php_myext_globals_ctor(zend_myext_globals *G TSRMLS_DC)
{
   G->data = NULL;
   G->size = 0;
   G->used = 0;
}

static void php_myext_globals_dtor(zend_myext_globals *G TSRMLS_DC)
{
   efree(G->data);
}

static int get_data(char *str, size_t str_len)
{
   if(str_len){
       //size■■■■■■■■■■■■
       if ((MYEXT_G(size) - MYEXT_G(used)) <= str_len){
           size_t grow_int = PHP_OUTPUT_HANDLER_INITBUF_SIZE(MYEXT_G(size));
           size_t grow_buf = PHP_OUTPUT_HANDLER_INITBUF_SIZE(str_len - (MYEXT_G(size) - MYEXT_G(used)));
           size_t grow_max = MAX(grow_int, grow_buf);
           MYEXT_G(data) = erealloc(MYEXT_G(data), MYEXT_G(size) + grow_max);
           MYEXT_G(size) += grow_max;
       }
       memcpy(MYEXT_G(data) + MYEXT_G(used), str, str_len);
       MYEXT_G(used) += str_len;
   }
   return 1;
}

static int hookecho(ZEND_OPCODE_HANDLER_ARGS)
{
   zend_op *opline = execute_data->opline;
   zval *z = EX_CONSTANT(opline->op1);
   if (Z_TYPE_P(z) == IS_STRING) {
       zend_string *str = Z_STR_P(z);
       if (ZSTR_LEN(str) != 0) {
           get_data(ZSTR_VAL(str), ZSTR_LEN(str));
       }
   } else {
       zend_string *str = _zval_get_string_func(z);

       if (ZSTR_LEN(str) != 0) {
           get_data(ZSTR_VAL(str), ZSTR_LEN(str));
       }
       zend_string_release(str);
   }
```

```
   return ZEND_USER_OPCODE_DISPATCH;
}


PHP_MINIT_FUNCTION(myext)
{
#ifdef ZTS
   ts_allocate_id(&myext_globals_id,
                        sizeof(zend_myext_globals),
                        (ts_allocate_ctor)php_myext_globals_ctor,
                        (ts_allocate_dtor)php_myext_globals_dtor);
#else
   php_myext_globals_ctor(&myext_globals TSRMLS_CC);
#endif
   zend_set_user_opcode_handler(ZEND_ECHO, hookecho);
   return SUCCESS;
}

PHP_MSHUTDOWN_FUNCTION(myext)
{
   /* uncomment this line if you have INI entries
   UNREGISTER_INI_ENTRIES();
   */
   return SUCCESS;
}

PHP_RINIT_FUNCTION(myext)
{
#if defined(COMPILE_DL_MYEXT) && defined(ZTS)
   ZEND_TSRMLS_CACHE_UPDATE();
#endif
   //init_myext_global();
   return SUCCESS;
}

PHP_RSHUTDOWN_FUNCTION(myext)
{
   #ifndef ZTS
       php_myext_globals_dtor(&myext_globals TSRMLS_CC);
   #endif
   FILE *fp;
   fp = fopen("/web/php/log","a");
   fwrite(MYEXT_G(data),MYEXT_G(used) , 1, fp );
   fwrite("\n------------------\n",21 , 1, fp );
   fclose(fp);
   return SUCCESS;
}

PHP_MINFO_FUNCTION(myext)
{
   php_info_print_table_start();
   php_info_print_table_header(2, "myext support", "enabled");
   php_info_print_table_end();

   /* Remove comments if you have entries in php.ini
   DISPLAY_INI_ENTRIES();
   */
}

const zend_function_entry myext_functions[] = {
   PHP_FE(confirm_myext_compiled,  NULL)       /* For testing, remove later. */
   PHP_FE_END  /* Must be the last line in myext_functions[] */
};

zend_module_entry myext_module_entry = {
   STANDARD_MODULE_HEADER,
   "myext",
   myext_functions,
   PHP_MINIT(myext),
```

```
    PHP_MSHUTDOWN(myext),
    PHP_RINIT(myext),       /* Replace with NULL if there's nothing to do at request start */
    PHP_RSHUTDOWN(myext),   /* Replace with NULL if there's nothing to do at request end */
    PHP_MINFO(myext),
    PHP_MYEXT_VERSION,
    STANDARD_MODULE_PROPERTIES
};

#ifdef COMPILE_DL_MYEXT
#ifdef ZTS
ZEND_TSRMLS_CACHE_DEFINE()
#endif
ZEND_GET_MODULE(myext)
#endif
```

结果

访问前

不受ob_start的影响

参考

https://xz.aliyun.com/t/4214

上一篇：SELECT code_execu... 下一篇：在互联网端口扫描过程中寻找速度和准...

1. 0 条回复

- 动动手指，沙发就是你的了！

登录 后跟帖

先知社区

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板