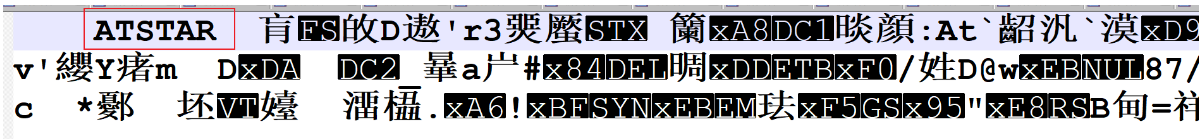
0x01 解密分析

需要解密的代码长大概这个样子



```
    ATSTAR   肓FS敀D遬'r3褜靨STX 簢xA8DC1晱顔:At`韶汎`漠xD9
v'纓Y瘏m  DxDA  DC2 暈a户#x84DEL啁xDDETBxF0/姓D@wxEBNUL87/
c  *鄹  坯VT嬞  澑樞.xA6!xBFSYNxEBEM玞xF5GSx95"xE8RSB甸=袺
```

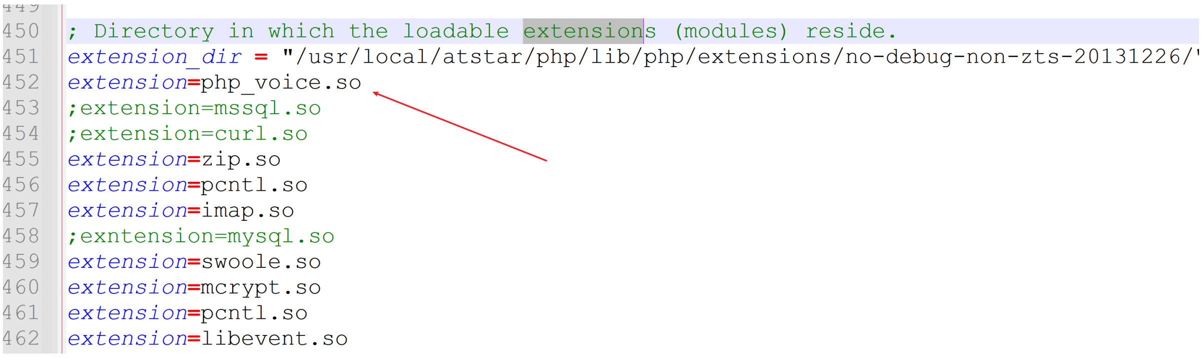
可以发现前面有个ATSTAR关键字，然后根据这个关键字搜索网上公开的加密方法

https://paper.seebug.org/478/

最初是找到了这篇文章，然后获取了一些解密的思路和方法。

但是文章没有提供解密脚本。

然后通过排查对应源码的php.ini文件，获取了可能存在的extension



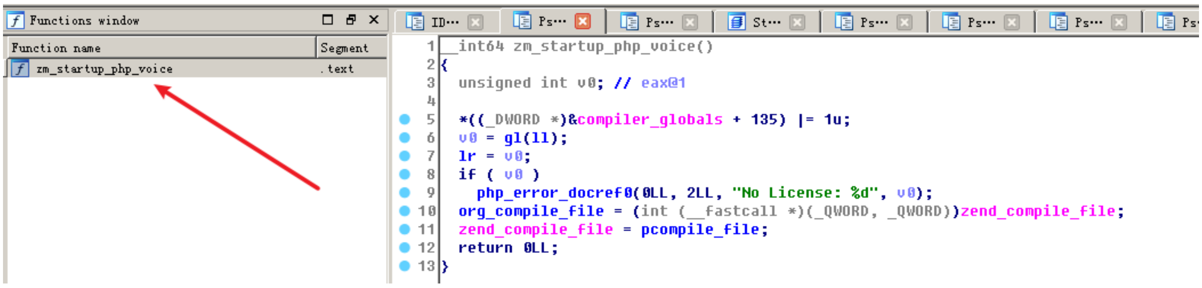
```
; Directory in which the loadable extensions (modules) reside.
extension_dir = "/usr/local/atstar/php/lib/php/extensions/no-debug-non-zts-20131226/'
extension=php_voice.so
;extension=mssql.so
;extension=curl.so
extension=zip.so
extension=pcntl.so
extension=imap.so
;exntension=mysql.so
extension=swoole.so
extension=mcrypt.so
extension=pcntl.so
extension=libevent.so
```

这里排除的方法一个是通过加密的名字，因为这个加密的名字就叫做voicestar

另一个是排除其他extension的so文件，因为根据名字可以查到其他so文件的用途

那么ok，现在就找到了这个so文件，进入逆向的过程

首先打开ida

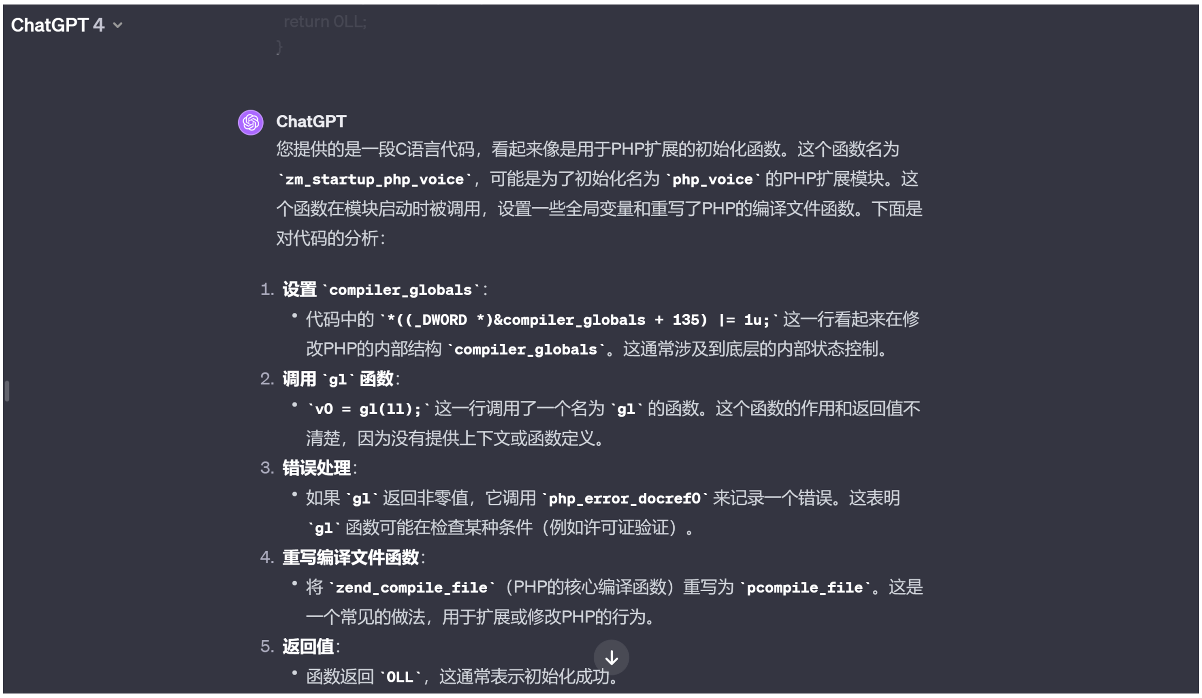


```
__int64 zm_startup_php_voice()
{
  unsigned int v0; // eax@1

  *((_DWORD *)&compiler_globals + 135) |= 1u;
  v0 = gl(11);
  lr = v0;
  if ( v0 )
    php_error_docref0(0LL, 2LL, "No License: %d", v0);
  org_compile_file = (int (__fastcall *)(_QWORD, _QWORD))zend_compile_file;
  zend_compile_file = pcompile_file;
  return 0LL;
}
```

在这里找到文章是说的入口函数，然后跟进

其实前半段内容基本和文章重复，唯一的不同在于现在工具更加先进了。

文章还需要自己看代码，我是先用ida的f5换成高级语言，然后直接把这段东西拷贝到gpt里面去。



```
    return 0LL;
}
```

ChatGPT

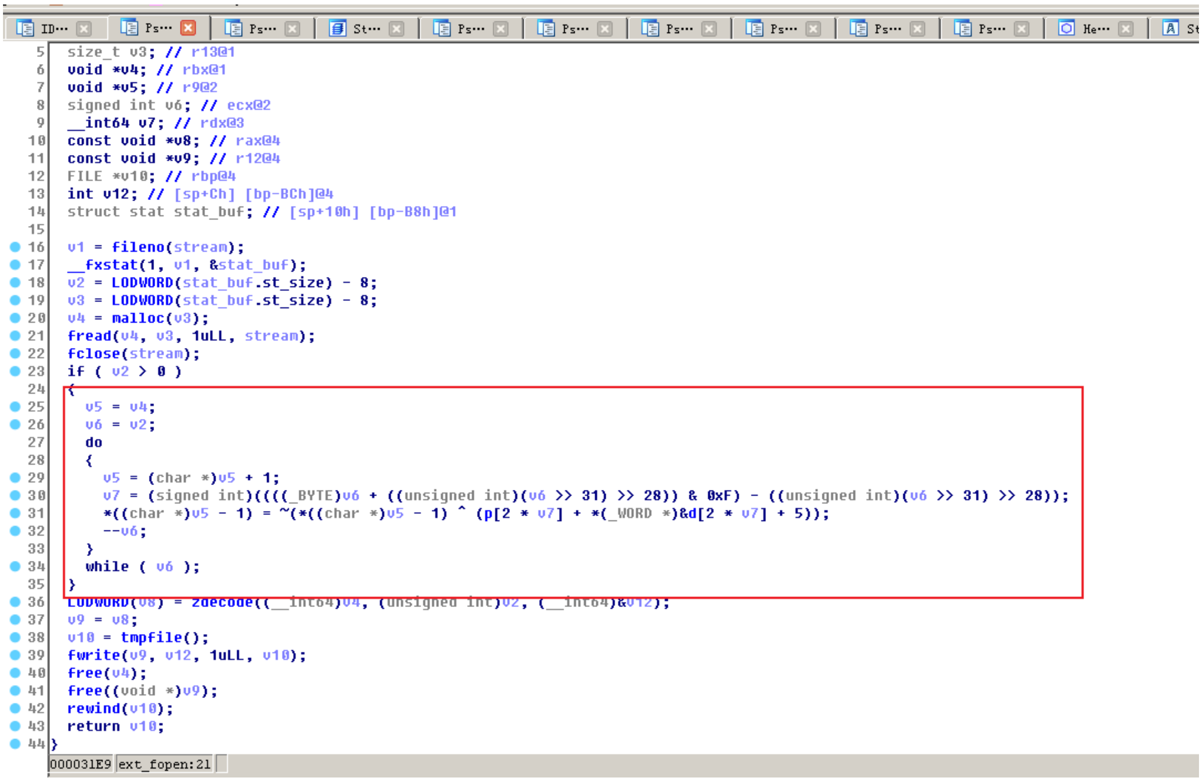
您提供的是一段C语言代码，看起来像是用于PHP扩展的初始化函数。这个函数名为 `zm_startup_php_voice`，可能是为了初始化名为 `php_voice` 的PHP扩展模块。这个函数在模块启动时被调用，设置一些全局变量和重写了PHP的编译文件函数。下面是对代码的分析：

1. **设置 `compiler_globals`**：
   - 代码中的 `*((_DWORD *)&compiler_globals + 135) |= 1u;` 这一行看起来在修改PHP的内部结构 `compiler_globals`。这通常涉及到底层的内部状态控制。
2. **调用 `gl` 函数**：
   - `v0 = gl(11);` 这一行调用了一个名为 `gl` 的函数。这个函数的作用和返回值不清楚，因为没有提供上下文或函数定义。
3. **错误处理**：
   - 如果 `gl` 返回非零值，它调用 `php_error_docref0` 来记录一个错误。这表明 `gl` 函数可能在检查某种条件（例如许可证验证）。
4. **重写编译文件函数**：
   - 将 `zend_compile_file`（PHP的核心编译函数）重写为 `pcompile_file`。这是一个常见的做法，用于扩展或修改PHP的行为。
5. **返回值**：
   - 函数返回 `0LL`，这通常表示初始化成功。

让他直接翻译，看都懒得看。

其他上文作者已经写过的，我就不写了，我补充以下作者没有写出来的。

首先还是回到代码中。



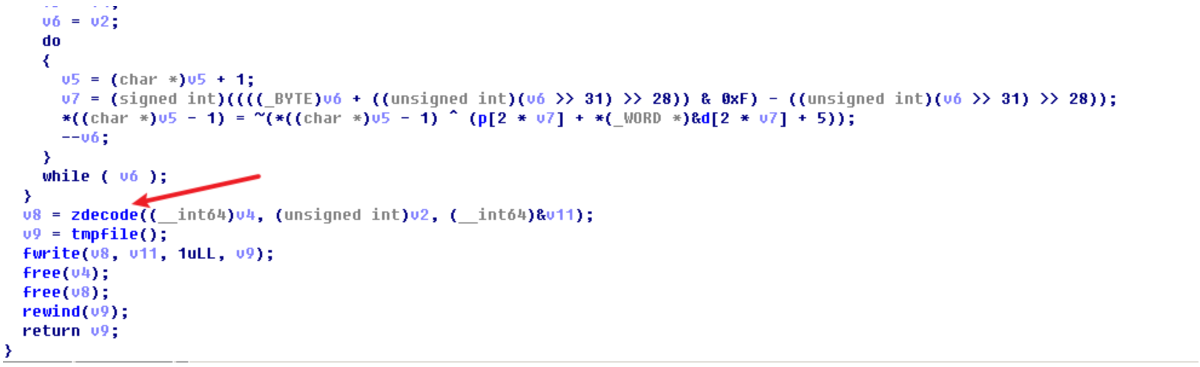
```
  size_t v3; // r13@1
  void *v4; // rbx@1
  void *v5; // r9@2
  signed int v6; // ecx@2
  __int64 v7; // rdx@3
  const void *v8; // rax@4
  const void *v9; // r12@4
  FILE *v10; // rbp@4
  int v12; // [sp+Ch] [bp-BCh]@4
  struct stat stat_buf; // [sp+10h] [bp-B8h]@1

  v1 = fileno(stream);
  __fxstat(1, v1, &stat_buf);
  v2 = LODWORD(stat_buf.st_size) - 8;
  v3 = LODWORD(stat_buf.st_size) - 8;
  v4 = malloc(v3);
  fread(v4, v3, 1uLL, stream);
  fclose(stream);
  if ( v2 > 0 )
  {
    v5 = v4;
    v6 = v2;
    do
    {
      v5 = (char *)v5 + 1;
      v7 = (signed int)((((_BYTE)v6 + ((unsigned int)(v6 >> 31) >> 28)) & 0xF) - ((unsigned int)(v6 >> 31) >> 28));
      *((char *)v5 - 1) = ~(*((char *)v5 - 1) ^ (p[2 * v7] + *(_WORD *)&d[2 * v7] + 5));
      --v6;
    }
    while ( v6 );
  }
  LODWORD(v8) = zdecode((__int64)v4, (unsigned int)v2, (__int64)&v12);
  v9 = v8;
  v10 = tmpfile();
  fwrite(v9, v12, 1uLL, v10);
  free(v4);
  free((void *)v9);
  rewind(v10);
  return v10;
}
```

这里因为已经有前辈帮我们找到了解密的过程。

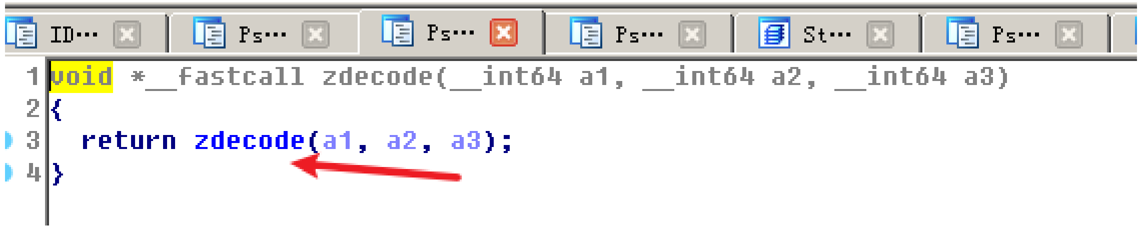
首先是这里的一个对称加密解密。

然后再是下面的一个zdecode

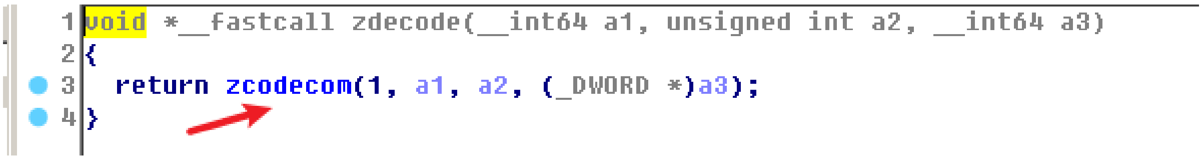
下面我写一下跟进zdecode的过程



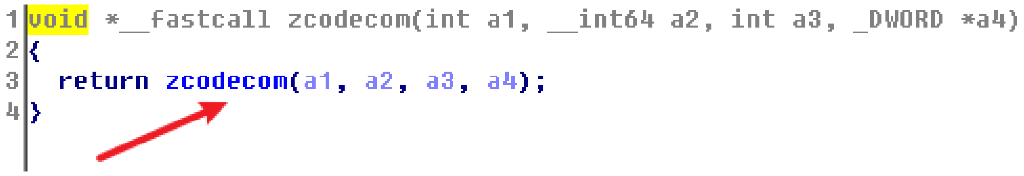
```
    v6 = v2;
    do
    {
      v5 = (char *)v5 + 1;
      v7 = (signed int)((((_BYTE)v6 + ((unsigned int)(v6 >> 31) >> 28)) & 0xF) - ((unsigned int)(v6 >> 31) >> 28));
      *((char *)v5 - 1) = ~(*((char *)v5 - 1) ^ (p[2 * v7] + *(_WORD *)&d[2 * v7] + 5));
      --v6;
    }
    while ( v6 );
  }
  v8 = zdecode((__int64)v4, (unsigned int)v2, (__int64)&v11);
  v9 = tmpfile();
  fwrite(v8, v11, 1uLL, v9);
  free(v4);
  free(v8);
  rewind(v9);
  return v9;
}
```



```
void *__fastcall zdecode(__int64 a1, __int64 a2, __int64 a3)
{
  return zdecode(a1, a2, a3);
}
```



```
void *__fastcall zdecode(__int64 a1, unsigned int a2, __int64 a3)
{
  return zcodecom(1, a1, a2, (_DWORD *)a3);
}
```



```
void *__fastcall zcodecom(int a1, __int64 a2, int a3, _DWORD *a4)
{
  return zcodecom(a1, a2, a3, a4);
}
```

```
void *__fastcall zcodecom(int a1, __int64 a2, int a3, _DWORD *a4)
{
  _DWORD *v4; // r15@1
  int v5; // ebp@1
  int v6; // ebx@1
  __int64 v7; // rbp@3
  void *v8; // r12@3
  int v9; // er14@4
  int v10; // eax@5
  int v11; // ebp@10
  void *v12; // rdi@11
  char *v14; // rax@20
  char *v15; // rdi@20
  char *v16; // rax@22
  char *v17; // rdi@22

  v4 = a4;
  v5 = a3;
  v6 = a1;
  *((_QWORD *)&z + 8) = 0LL;
  *((_QWORD *)&z + 9) = 0LL;
  *((_QWORD *)&z + 10) = 0LL;
  z = 0LL;
  *((_DWORD *)&z + 2) = 0;
  if ( a1 )
    inflateInit_(&z, "1.2.8", 112LL);
  else
    deflateInit_(&z, 1LL, "1.2.8", 112LL);
  z = a2;
  *((_DWORD *)&z + 2) = v5;
  *((_DWORD *)&z + 8) = 100000;
  v7 = 0LL;
  *((_QWORD *)&z + 3) = &outbuf;
  v8 = malloc(0x186A0uLL);
LABEL_4:
  v9 = v7;
  while ( 1 )
  {
    if ( v6 )
    {
      v10 = inflate(&z, 0LL);
      if ( v10 == 1 )
      {
LABEL_10:
        v11 = 100000 - *((_DWORD *)&z + 8);
        if ( 100000 == *((_DWORD *)&z + 8) )
        {
          v12 = &z;
          if ( v6 )
            goto LABEL_12;
LABEL_16:
          deflateEnd(v12);
        }
        else
        {
          v16 = (char *)realloc(v8, v9 + 100000);
          v17 = &v16[v9];
          v8 = v16;
          v9 += v11;
          memcpy(v17, &outbuf, v11);
          v12 = &z;
          if ( !v6 )
            goto LABEL_16;
LABEL_12:
          inflateEnd(v12);
        }
        *v4 = v9;
        return v8;
      }
    }
    else
    {
      v10 = deflate(&z, 4LL);
      if ( v10 == 1 )
        goto LABEL_10;
    }
    if ( v10 )
      break;
    if ( !*((_DWORD *)&z + 8) )
    {
      v14 = (char *)realloc(v8, v7 + 100000);
      v15 = &v14[v7];
      v8 = v14;
      v7 += 100000LL;
      memcpy(v15, &outbuf, 0x186A0uLL);
      *((_DWORD *)&z + 8) = 100000;
      *((_QWORD *)&z + 3) = &outbuf;
      goto LABEL_4;
    }
  }
  if ( v6 )
    inflateEnd(&z);
  else
    deflateEnd(&z);
  *v4 = 0;
  return v8;
}
```

可以看到，zdecode本质是解压的过程。

然后解压这块，原文作者已经提供了一种实现方法，就是用python来做解压，这里我们先不管。

然后就到了前面的解密部分，因为他没把脚本放出来，于是我就开始写脚本。

这里我有两种思路

0x02 解密实现

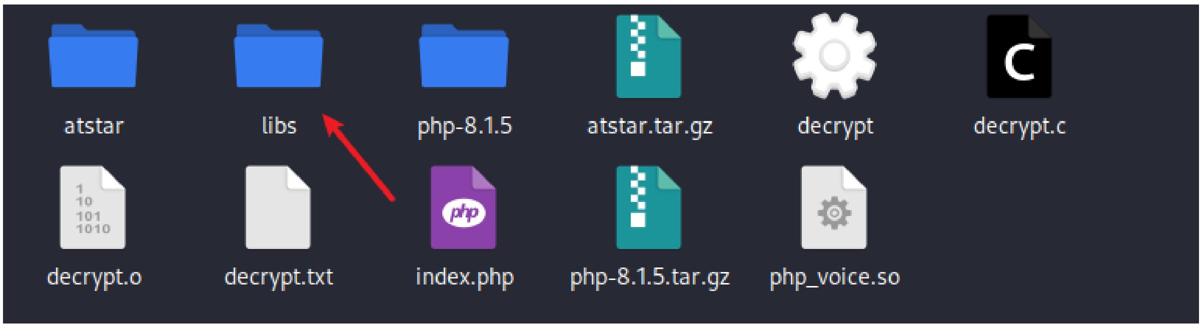
思路一 调用so文件中的解密函数来解密

这里我用c实现了一个脚本，来加载这个so文件，然后调用其中的解密函数直接实现解密

```
#include <stdio.h>
#include <stdlib.h>
#include <dlfcn.h>

// 定义函数指针类型
typedef FILE* (*ext_fopen_func)(FILE*);

int main() {
    void* handle;
    char* error;
    ext_fopen_func ext_fopen;

    // 打开动态库
    handle = dlopen("./php_voice.so", RTLD_LAZY);
    if (!handle) {
        fprintf(stderr, "%s\n", dlerror());
        exit(EXIT_FAILURE);
    }

    // 获取函数指针
    ext_fopen = (ext_fopen_func)dlsym(handle, "ext_fopen");
    if ((error = dlerror()) != NULL) {
        fprintf(stderr, "%s\n", error);
        exit(EXIT_FAILURE);
    }

    // 打开加密文件
    FILE* encrypted_file = fopen("./index.php", "rb");
    if (encrypted_file == NULL) {
        perror("Error opening file");
        exit(EXIT_FAILURE);
    }

    // 调用 ext_fopen 函数解密
    FILE* decrypted_file = ext_fopen(encrypted_file);

    // 打开输出文件
    FILE* output_file = fopen("./decrypt.txt", "w");
    if (output_file == NULL) {
        perror("Error opening output file");
        exit(EXIT_FAILURE);
    }

    // 从解密文件中读取数据并写入到输出文件
    char buffer[1024];
    size_t bytes_read;
    while ((bytes_read = fread(buffer, 1, sizeof(buffer), decrypted_file)) > 0) {
        fwrite(buffer, 1, bytes_read, output_file);
    }

    // 清理工作
    fclose(output_file);
    fclose(decrypted_file);  // 假设 ext_fopen 不关闭原始文件流
    dlclose(handle);

    return 0;
}
```

脚本执行需要满足一些条件

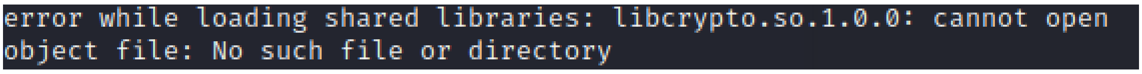
1.需要引入对应的lib库，这里我把lib放进来了。



然后在bash上面引入环境变量

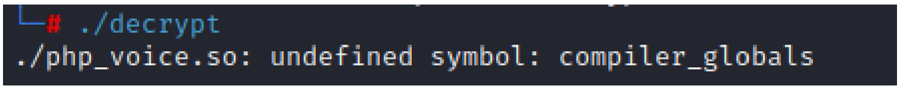
```
export LD_LIBRARY_PATH=$(pwd)/libs:$LD_LIBRARY_PATH
```

即可运行，否则会报错缺少库



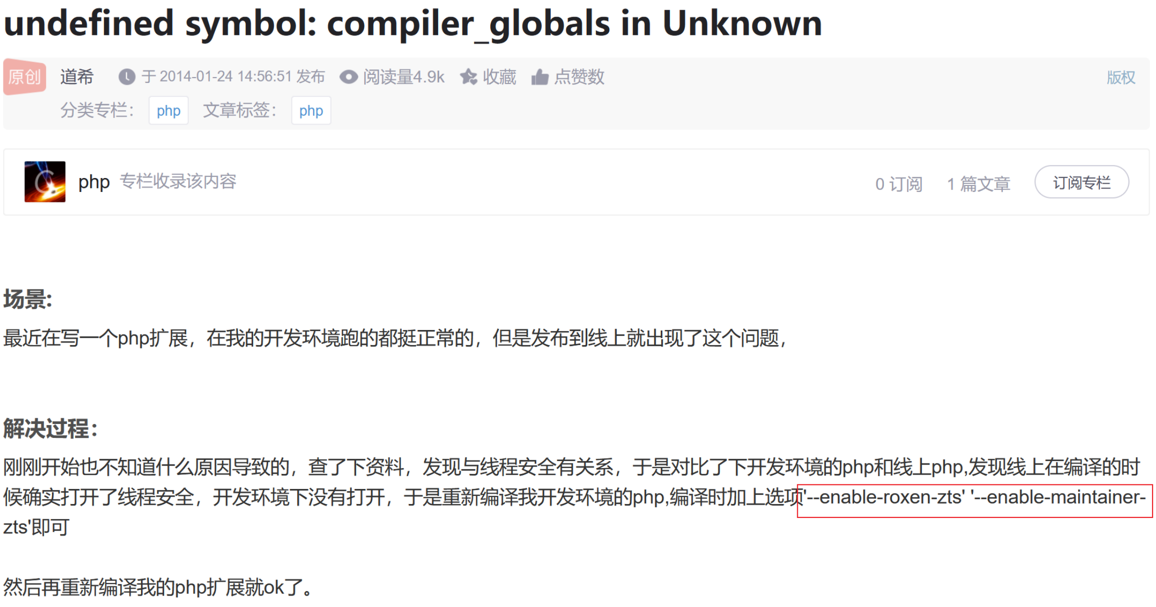
```
error while loading shared libraries: libcrypto.so.1.0.0: cannot open
object file: No such file or directory
```

然后库环境搭建好了之后，又会遇到另一个问题，就是



```
└─# ./decrypt
./php_voice.so: undefined symbol: compiler_globals
```

这个问题百度查询后，发现是由于本地没有开启php的线程安全选项导致的



undefined symbol: compiler_globals in Unknown

原创 道希 于 2014-01-24 14:56:51 发布 阅读量4.9k 收藏 点赞数 版权
分类专栏： php 文章标签： php

php 专栏收录该内容 0 订阅 1 篇文章 订阅专栏

场景:

最近在写一个php扩展，在我的开发环境跑的都挺正常的，但是发布到线上就出现了这个问题，

解决过程:

刚刚开始也不知道什么原因导致的，查了下资料，发现与线程安全有关系，于是对比了下开发环境的php和线上php,发现线上在编译的时候确实打开了线程安全，开发环境下没有打开，于是重新编译我开发环境的php,编译时加上选项'--enable-roxen-zts' '--enable-maintainer-zts'即可

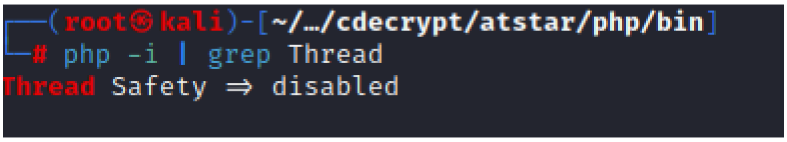
然后再重新编译我的php扩展就ok了。

这边直接找到文章

https://blog.csdn.net/KitrosMC/article/details/124524078

然后尝试用文章中的方法重新编译一个php

这里在我查看本机php版本的时候，发现线程安全确实没有开启

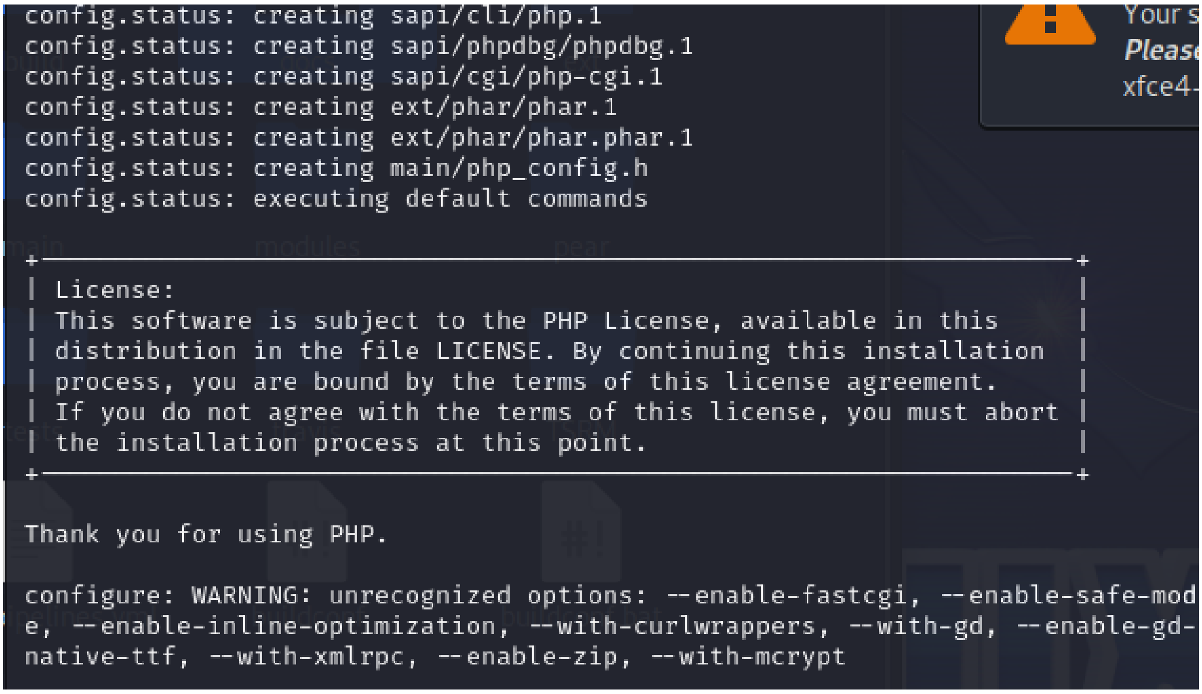


```
┌──(root㉿kali)-[~/…/cdecrypt/atstar/php/bin]
└─# php -i | grep Thread
Thread Safety => disabled
```

那么就尝试重新编译php后开启，再运行一下试试

这里编译坑是真的巨多，解决了多个依赖的问题之后，这里终于configure成功了。

```
sudo apt-get install sqlite3 libsqlite3-dev
sudo apt-get install libcurl4 libcurl4-openssl-dev
sudo apt-get install libonig-dev
```

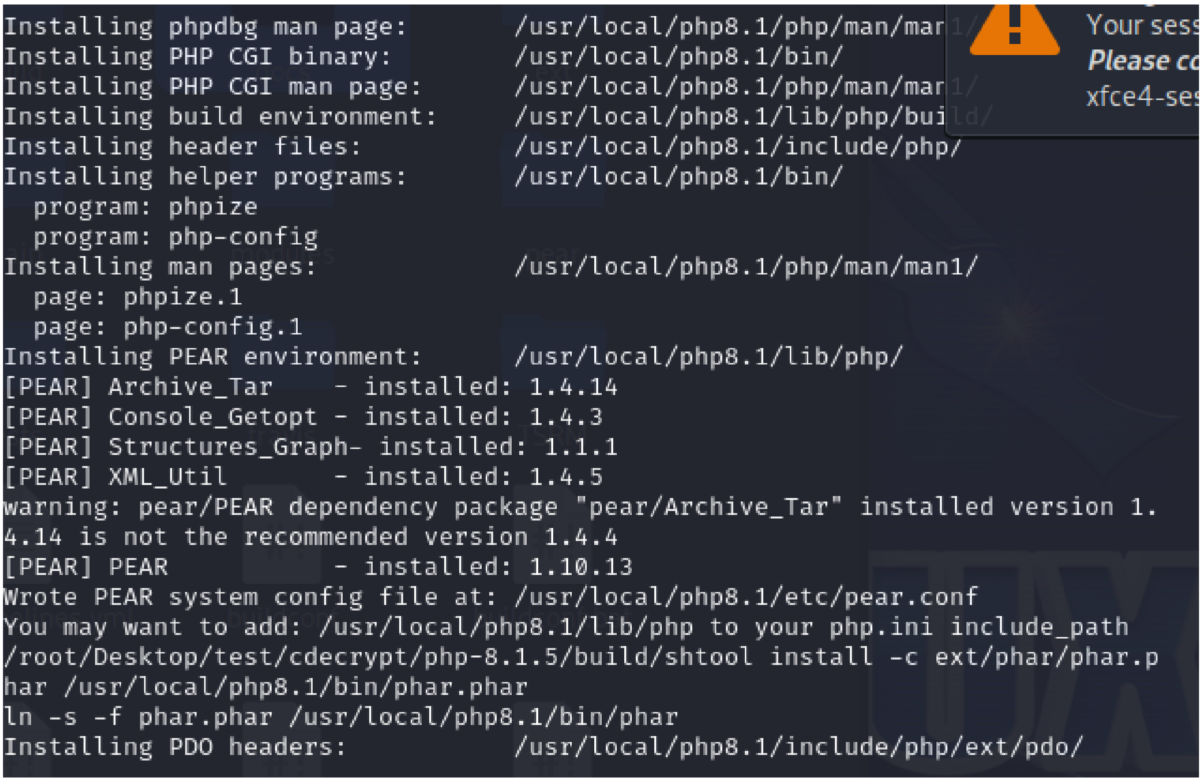


```
config.status: creating sapi/cli/php.1
config.status: creating sapi/phpdbg/phpdbg.1
config.status: creating sapi/cgi/php-cgi.1
config.status: creating ext/phar/phar.1
config.status: creating ext/phar/phar.phar.1
config.status: creating main/php_config.h
config.status: executing default commands

+--------------------------------------------------------------------+
| License:                                                           |
| This software is subject to the PHP License, available in this    |
| distribution in the file LICENSE. By continuing this installation |
| process, you are bound by the terms of this license agreement.    |
| If you do not agree with the terms of this license, you must abort |
| the installation process at this point.                            |
+--------------------------------------------------------------------+

Thank you for using PHP.

configure: WARNING: unrecognized options: --enable-fastcgi, --enable-safe-mod
e, --enable-inline-optimization, --with-curlwrappers, --with-gd, --enable-gd-
native-ttf, --with-xmlrpc, --enable-zip, --with-mcrypt
```

然后开始编译

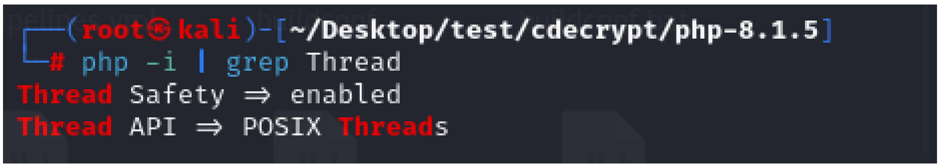
```
sudo make && make install
```

编译成功



```
Installing phpdbg man page:       /usr/local/php8.1/php/man/man1/
Installing PHP CGI binary:        /usr/local/php8.1/bin/
Installing PHP CGI man page:      /usr/local/php8.1/php/man/man1/
Installing build environment:     /usr/local/php8.1/lib/php/build/
Installing header files:          /usr/local/php8.1/include/php/
Installing helper programs:       /usr/local/php8.1/bin/
  program: phpize
  program: php-config
Installing man pages:             /usr/local/php8.1/php/man/man1/
  page: phpize.1
  page: php-config.1
Installing PEAR environment:      /usr/local/php8.1/lib/php/
[PEAR] Archive_Tar    - installed: 1.4.14
[PEAR] Console_Getopt - installed: 1.4.3
[PEAR] Structures_Graph- installed: 1.1.1
[PEAR] XML_Util       - installed: 1.4.5
warning: pear/PEAR dependency package "pear/Archive_Tar" installed version 1.
4.14 is not the recommended version 1.4.4
[PEAR] PEAR           - installed: 1.10.13
Wrote PEAR system config file at: /usr/local/php8.1/etc/pear.conf
You may want to add: /usr/local/php8.1/lib/php to your php.ini include_path
/root/Desktop/test/cdecrypt/php-8.1.5/build/shtool install -c ext/phar/phar.p
har /usr/local/php8.1/bin/phar.phar
ln -s -f phar.phar /usr/local/php8.1/bin/phar
Installing PDO headers:           /usr/local/php8.1/include/php/ext/pdo/
```

然后这里查看是否开启了线程安全



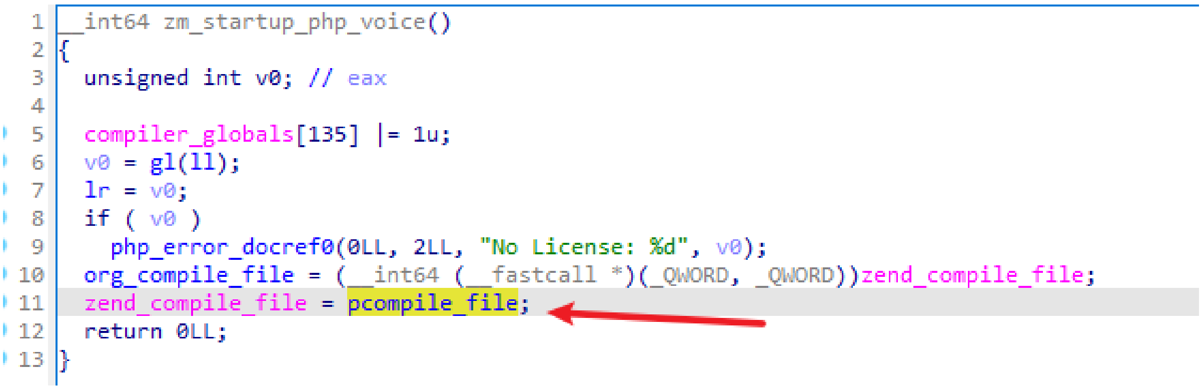
```
┌──(root㉿kali)-[~/Desktop/test/cdecrypt/php-8.1.5]
└─# php -i | grep Thread
Thread Safety => enabled
Thread API => POSIX Threads
```

然后尝试运行脚本，发现跑不起来，有点东西，直接g

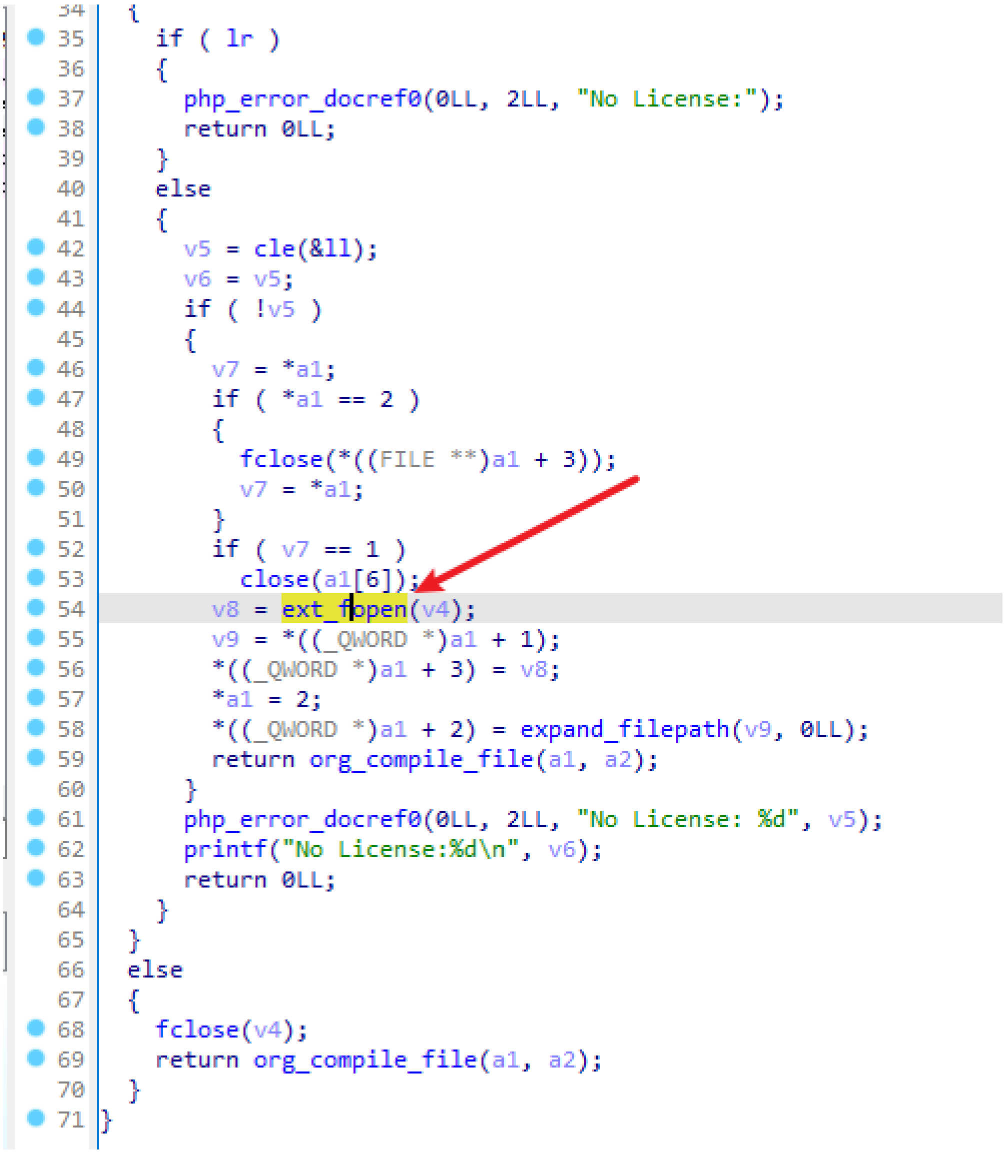
这里调试已经花费了很多时间了，有点难受，感觉可能路子有点问题

思路二 直接逆

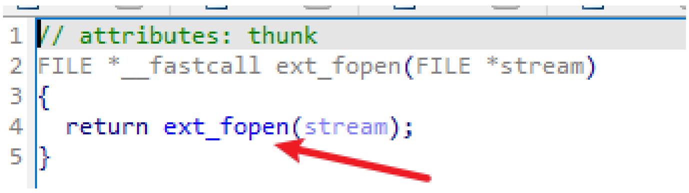
这里还是先完整跟一遍函数流动



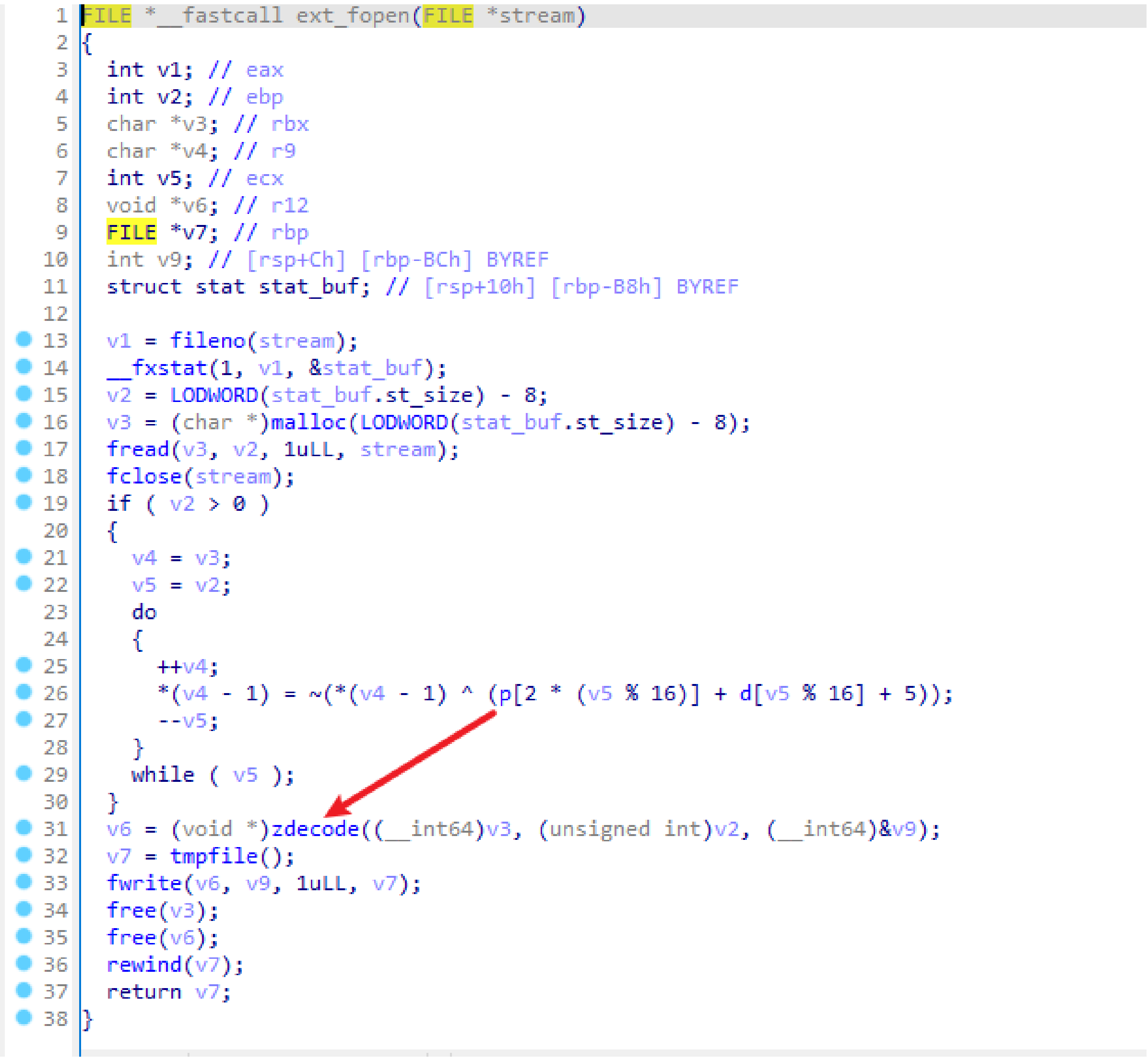
```
__int64 zm_startup_php_voice()
{
  unsigned int v0; // eax

  compiler_globals[135] |= 1u;
  v0 = gl(1LL);
  lr = v0;
  if ( v0 )
    php_error_docref0(0LL, 2LL, "No License: %d", v0);
  org_compile_file = (__int64 (__fastcall *)(_QWORD, _QWORD))zend_compile_file;
  zend_compile_file = pcompile_file;
  return 0LL;
}
```



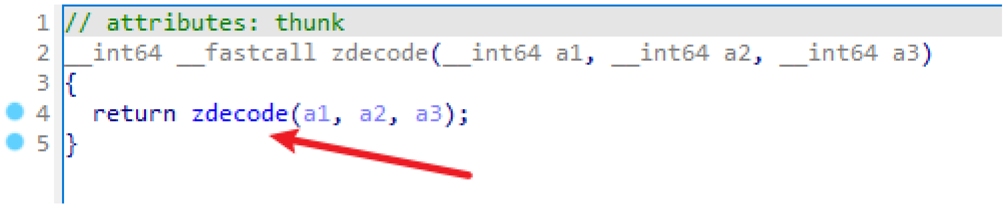
```
  {
    if ( lr )
    {
      php_error_docref0(0LL, 2LL, "No License:");
      return 0LL;
    }
    else
    {
      v5 = cle(&ll);
      v6 = v5;
      if ( !v5 )
      {
        v7 = *a1;
        if ( *a1 == 2 )
        {
          fclose(*((FILE **)a1 + 3));
          v7 = *a1;
        }
        if ( v7 == 1 )
          close(a1[6]);
        v8 = ext_fopen(v4);
        v9 = *((_QWORD *)a1 + 1);
        *((_QWORD *)a1 + 3) = v8;
        *a1 = 2;
        *((_QWORD *)a1 + 2) = expand_filepath(v9, 0LL);
        return org_compile_file(a1, a2);
      }
      php_error_docref0(0LL, 2LL, "No License: %d", v5);
      printf("No License:%d\n", v6);
      return 0LL;
    }
  }
  else
  {
    fclose(v4);
    return org_compile_file(a1, a2);
  }
}
```



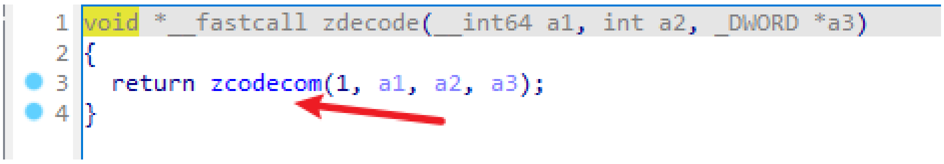
```
// attributes: thunk
FILE *__fastcall ext_fopen(FILE *stream)
{
  return ext_fopen(stream);
}
```



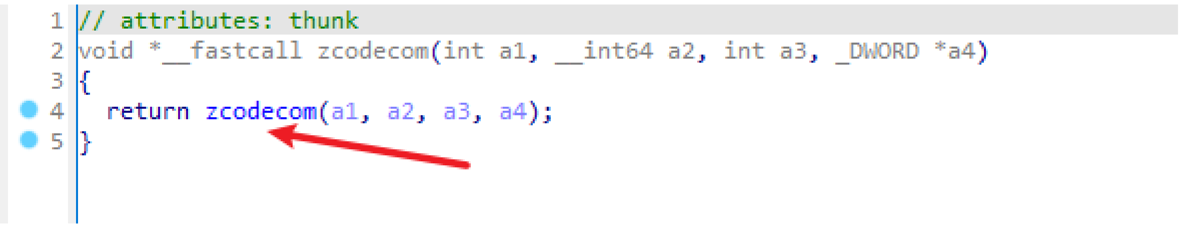
```
FILE *__fastcall ext_fopen(FILE *stream)
{
  int v1; // eax
  int v2; // ebp
  char *v3; // rbx
  char *v4; // r9
  int v5; // ecx
  void *v6; // r12
  FILE *v7; // rbp
  int v9; // [rsp+Ch] [rbp-BCh] BYREF
  struct stat stat_buf; // [rsp+10h] [rbp-B8h] BYREF

  v1 = fileno(stream);
  __fxstat(1, v1, &stat_buf);
  v2 = LODWORD(stat_buf.st_size) - 8;
  v3 = (char *)malloc(LODWORD(stat_buf.st_size) - 8);
  fread(v3, v2, 1uLL, stream);
  fclose(stream);
  if ( v2 > 0 )
  {
    v4 = v3;
    v5 = v2;
    do
    {
      ++v4;
      *(v4 - 1) = ~(*(v4 - 1) ^ (p[2 * (v5 % 16)] + d[v5 % 16] + 5));
      --v5;
    }
    while ( v5 );
  }
  v6 = (void *)zdecode((__int64)v3, (unsigned int)v2, (__int64)&v9);
  v7 = tmpfile();
  fwrite(v6, v9, 1uLL, v7);
  free(v3);
  free(v6);
  rewind(v7);
  return v7;
}
```



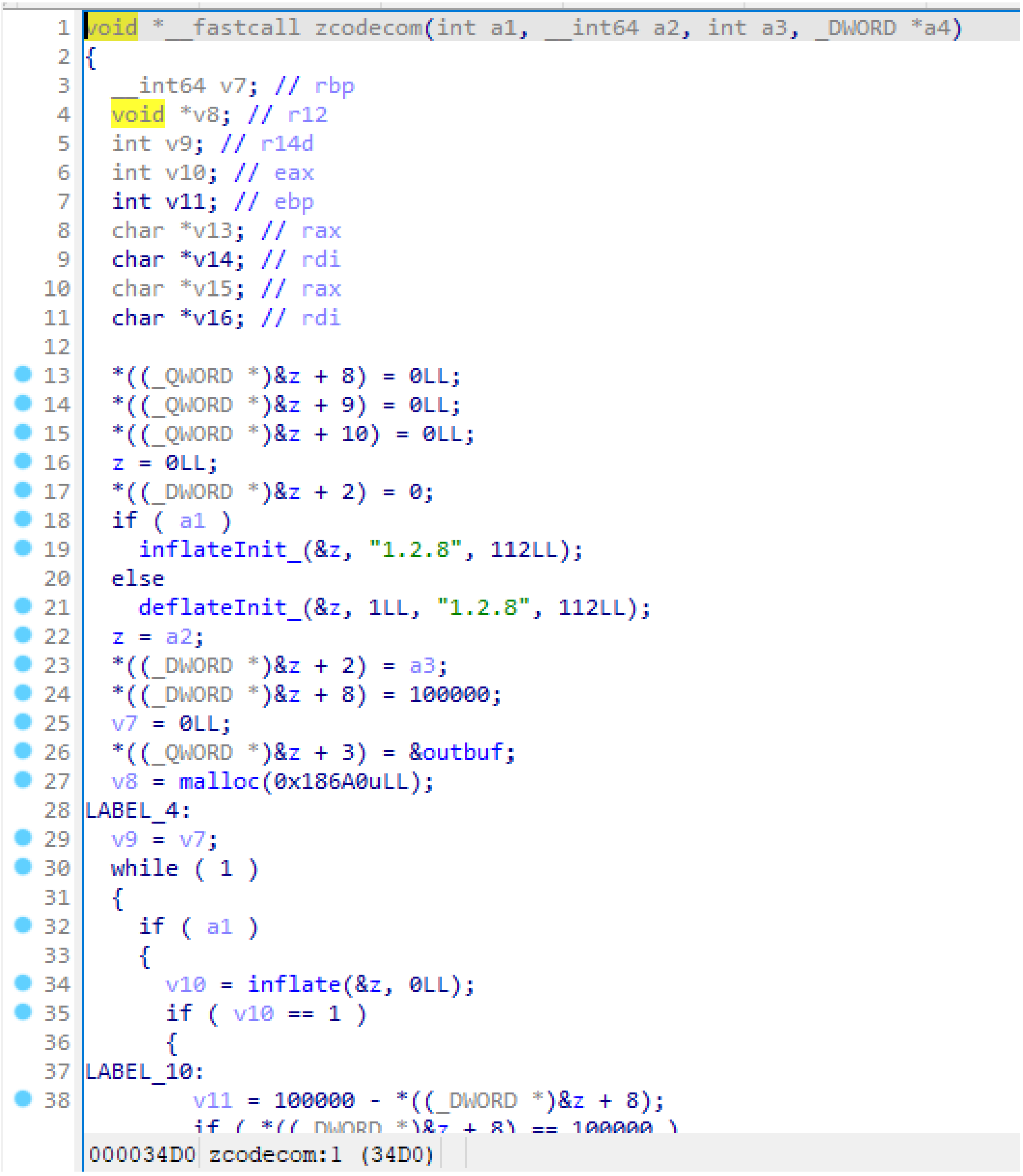
```
// attributes: thunk
__int64 __fastcall zdecode(__int64 a1, __int64 a2, __int64 a3)
{
  return zdecode(a1, a2, a3);
}
```



```
void *__fastcall zdecode(__int64 a1, int a2, _DWORD *a3)
{
  return zcodecom(1, a1, a2, a3);
}
```



```
// attributes: thunk
void *__fastcall zcodecom(int a1, __int64 a2, int a3, _DWORD *a4)
{
  return zcodecom(a1, a2, a3, a4);
}
```



```
void *__fastcall zcodecom(int a1, __int64 a2, int a3, _DWORD *a4)
{
  __int64 v7; // rbp
  void *v8; // r12
  int v9; // r14d
  int v10; // eax
  int v11; // ebp
  char *v13; // rax
  char *v14; // rdi
  char *v15; // rax
  char *v16; // rdi

  *((_QWORD *)&z + 8) = 0LL;
  *((_QWORD *)&z + 9) = 0LL;
  *((_QWORD *)&z + 10) = 0LL;
  z = 0LL;
  *((_DWORD *)&z + 2) = 0;
  if ( a1 )
    inflateInit_(&z, "1.2.8", 112LL);
  else
    deflateInit_(&z, 1LL, "1.2.8", 112LL);
  z = a2;
  *((_DWORD *)&z + 2) = a3;
  *((_DWORD *)&z + 8) = 100000;
  v7 = 0LL;
  *((_QWORD *)&z + 3) = &outbuf;
  v8 = malloc(0x186A0uLL);
LABEL_4:
  v9 = v7;
  while ( 1 )
  {
    if ( a1 )
    {
      v10 = inflate(&z, 0LL);
      if ( v10 == 1 )
      {
LABEL_10:
        v11 = 100000 - *((_DWORD *)&z + 8);
        if ( *((_DWORD *)&z + 8) == 100000 )
```

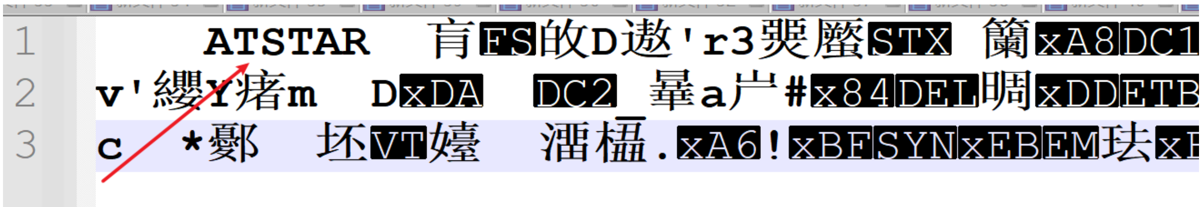
000034D0 zcodecom:1 (34D0)

然后按照对应思路路子编写解密脚本即可

关于https://paper.seebug.org/478/

这篇文章，其实他还有个点没有提到

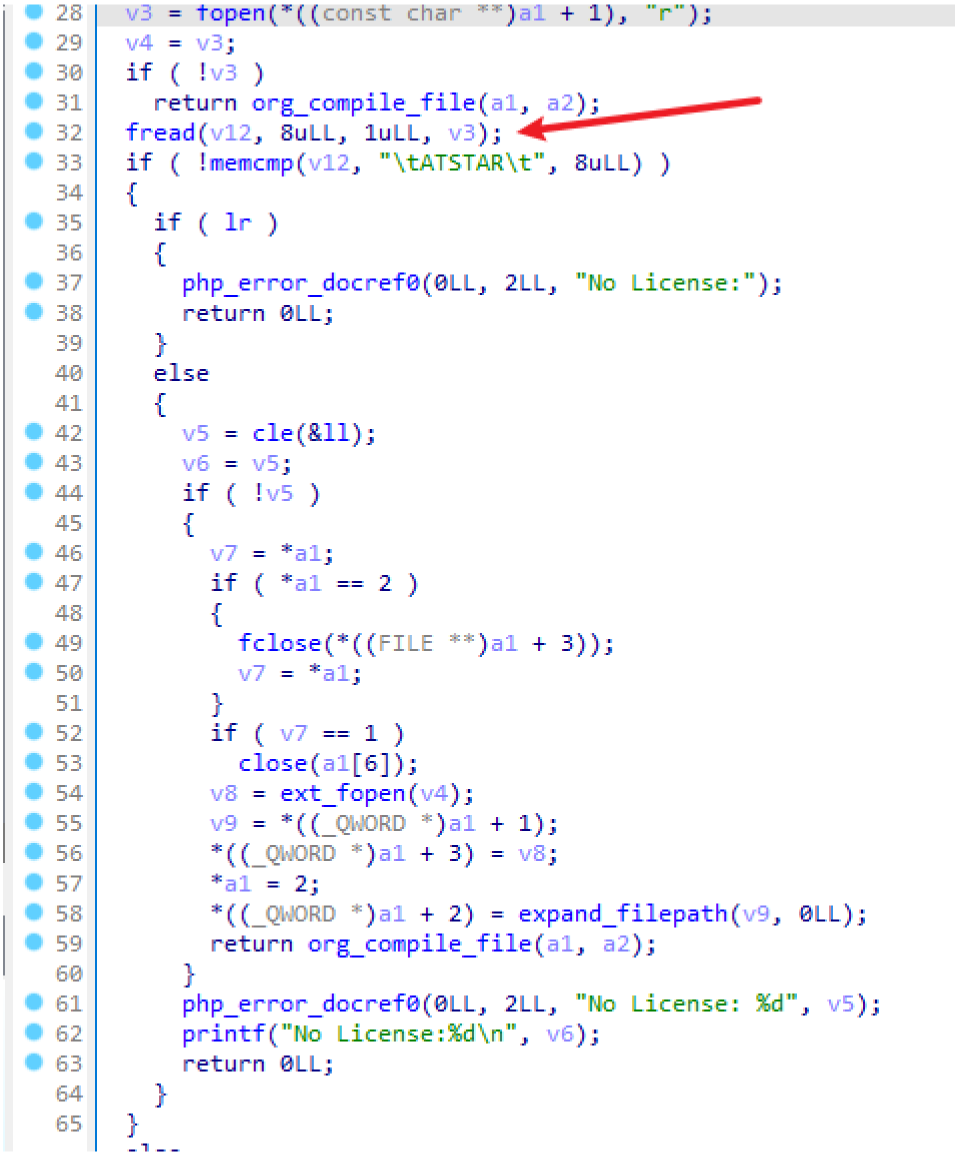
我们看到的加密后的代码是这样的



```
     ATSTAR  肓FS敀D遨'r3翜靨STX  籣xA8DC1
v'纓Y瘏m  DxDA  DC2  暈a户#x84DEL啁xDDETB
c   *鄹  坯VT嬄  漕楅.xA6!xBFSYNxEBEM珐xE
```

这前面一截东西，在解密的时候需要去掉的

在IDA中体现在这里



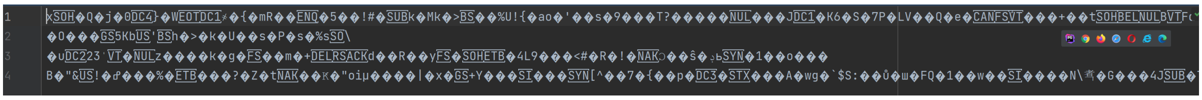
```
v3 = fopen(*((const char **)a1 + 1), "r");
v4 = v3;
if ( !v3 )
  return org_compile_file(a1, a2);
fread(v12, 8uLL, 1uLL, v3);
if ( !memcmp(v12, "\tATSTAR\t", 8uLL) )
{
  if ( lr )
  {
    php_error_docref0(0LL, 2LL, "No License:");
    return 0LL;
  }
  else
  {
    v5 = cle(&ll);
    v6 = v5;
    if ( !v5 )
    {
      v7 = *a1;
      if ( *a1 == 2 )
      {
        fclose(*((FILE **)a1 + 3));
        v7 = *a1;
      }
      if ( v7 == 1 )
        close(a1[6]);
      v8 = ext_fopen(v4);
      v9 = *((_QWORD *)a1 + 1);
      *((_QWORD *)a1 + 3) = v8;
      *a1 = 2;
      *((_QWORD *)a1 + 2) = expand_filepath(v9, 0LL);
      return org_compile_file(a1, a2);
    }
    php_error_docref0(0LL, 2LL, "No License: %d", v5);
    printf("No License:%d\n", v6);
    return 0LL;
  }
}
```

这里可以看到，是先读了8个字节，然后再走下面的ext_fopen()逻辑

也就是说，前面的8个字节是需要去掉的，不然是无法正常完成解密

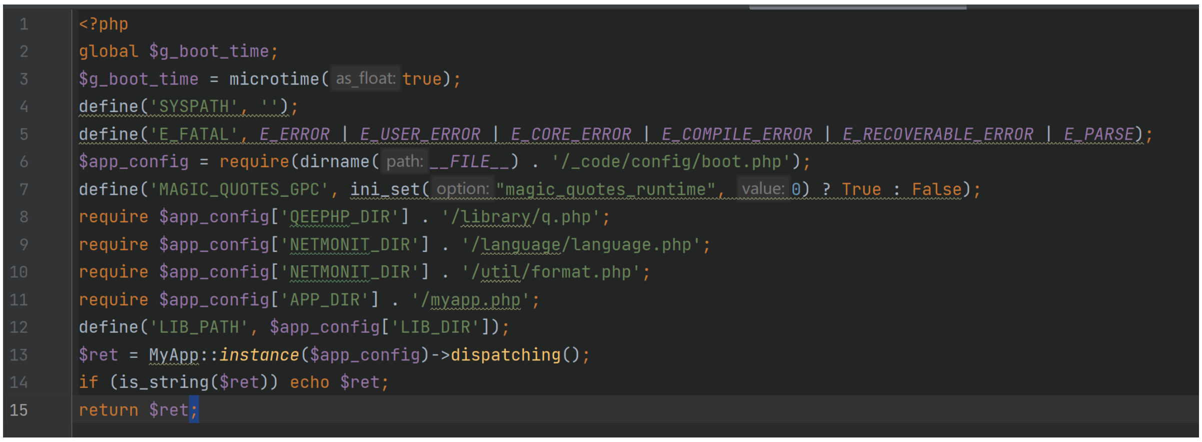
然后这里用脚本跑一下，就可以解密完成了

第一次异或解密后，跑出来是这样的



```
xSOH�Q�j�0DC4}�WEOTDC1≠�{�mR��ENQ�5��!#�SUBk�Mk�>BS��%U!{�ao�'��s�9���T?�����NUL���JDC1�K6�S�7P�LV��Q�e�CANFSVT���+��tSOHBELNULBVTF
�0���GS5KbUS'BSh�>�k�U��s�P�s�%sSO\
�uDC223'VT�NULz����k�g�FS��m�+DELRSACKd��R��yFS�SOHETB�4L9���<#�R�!�NAKↄ��ŝ�ⱼbSYN�1��o���
B�"&US!�ᵭ���%�ETB���?�Z�tNAK��ĸ�"oiµ����|�x�GS+Y���SI���SYN[^��7�{��p�DC3�STX���A�wg�`$S:��ů�ш�FQ�1��w��SI����N\耉�G���4JSUB�
```

然后再来一次py的解压，就可以获得明文了



```php
<?php
global $g_boot_time;
$g_boot_time = microtime( as_float: true);
define('SYSPATH', '');
define('E_FATAL', E_ERROR | E_USER_ERROR | E_CORE_ERROR | E_COMPILE_ERROR | E_RECOVERABLE_ERROR | E_PARSE);
$app_config = require(dirname( path: __FILE__) . '/_code/config/boot.php');
define('MAGIC_QUOTES_GPC', ini_set( option: "magic_quotes_runtime",  value: 0) ? True : False);
require $app_config['QEEPHP_DIR'] . '/library/q.php';
require $app_config['NETMONIT_DIR'] . '/language/language.php';
require $app_config['NETMONIT_DIR'] . '/util/format.php';
require $app_config['APP_DIR'] . '/myapp.php';
define('LIB_PATH', $app_config['LIB_DIR']);
$ret = MyApp::instance($app_config)->dispatching();
if (is_string($ret)) echo $ret;
return $ret;
```

然后接下来就可以开开心心的审计代码了

Done