Devlink
第二届PHP全球开发者大会

PHP 安全编码实战经验

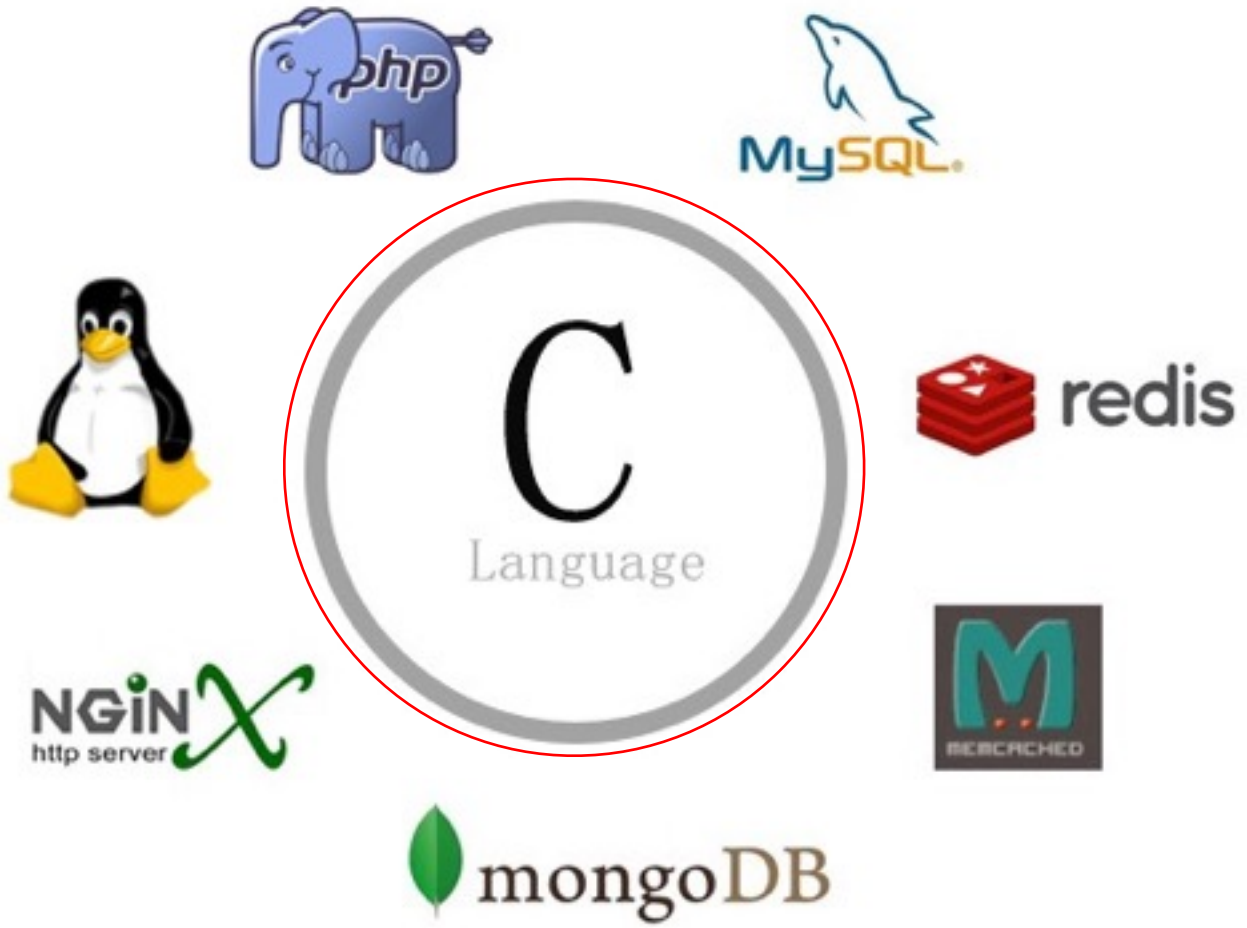
黄敏

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


php
MySQL
C
Language
redis
NGINX
http server
MEMCACHED
mongoDB

1.1 从 C 缓冲区溢出说起

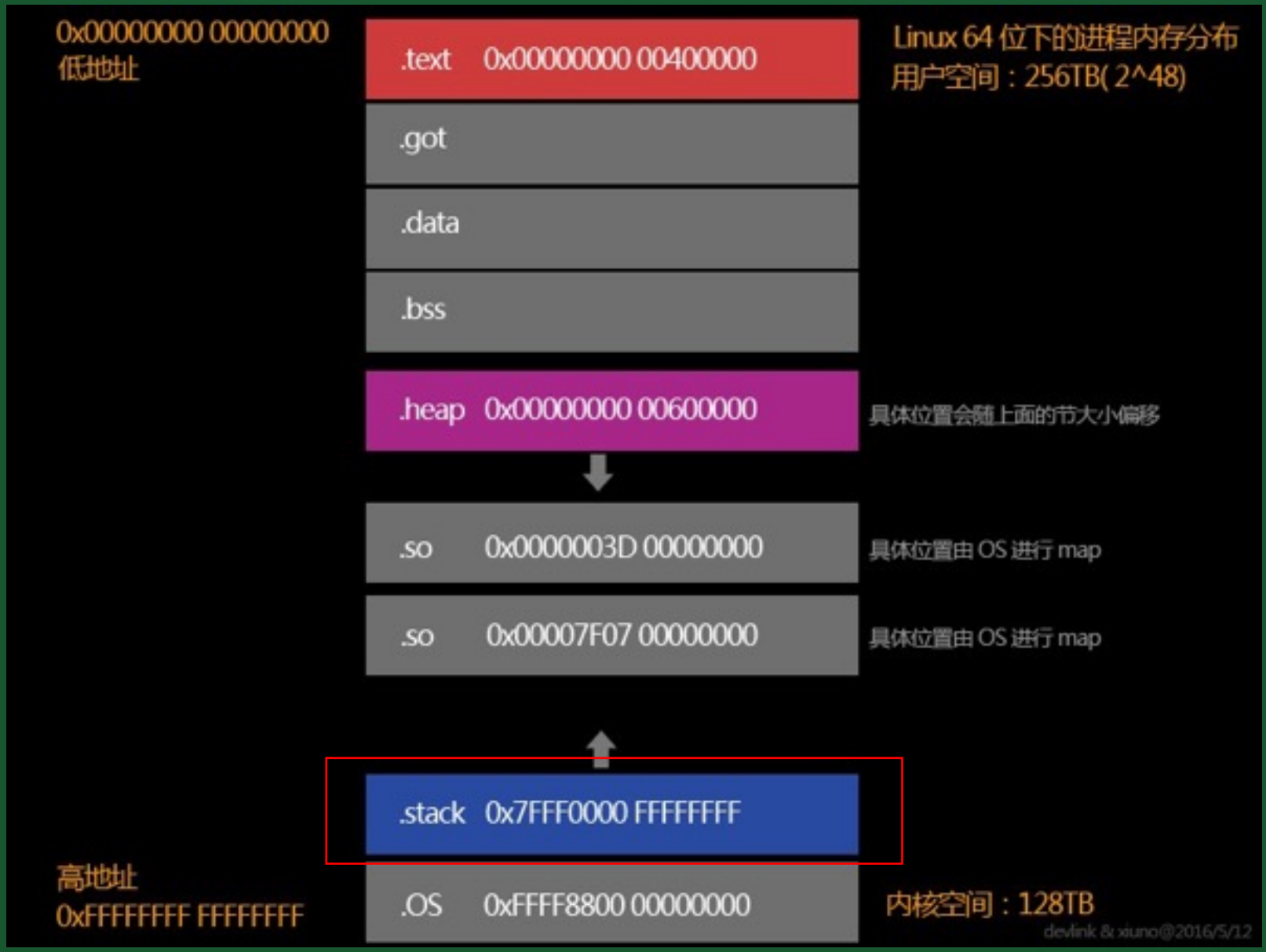
古老但将持续存在下去的问题

```c
#include <stdio.h>

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

int main() {

        func();
        return 0;

}
```

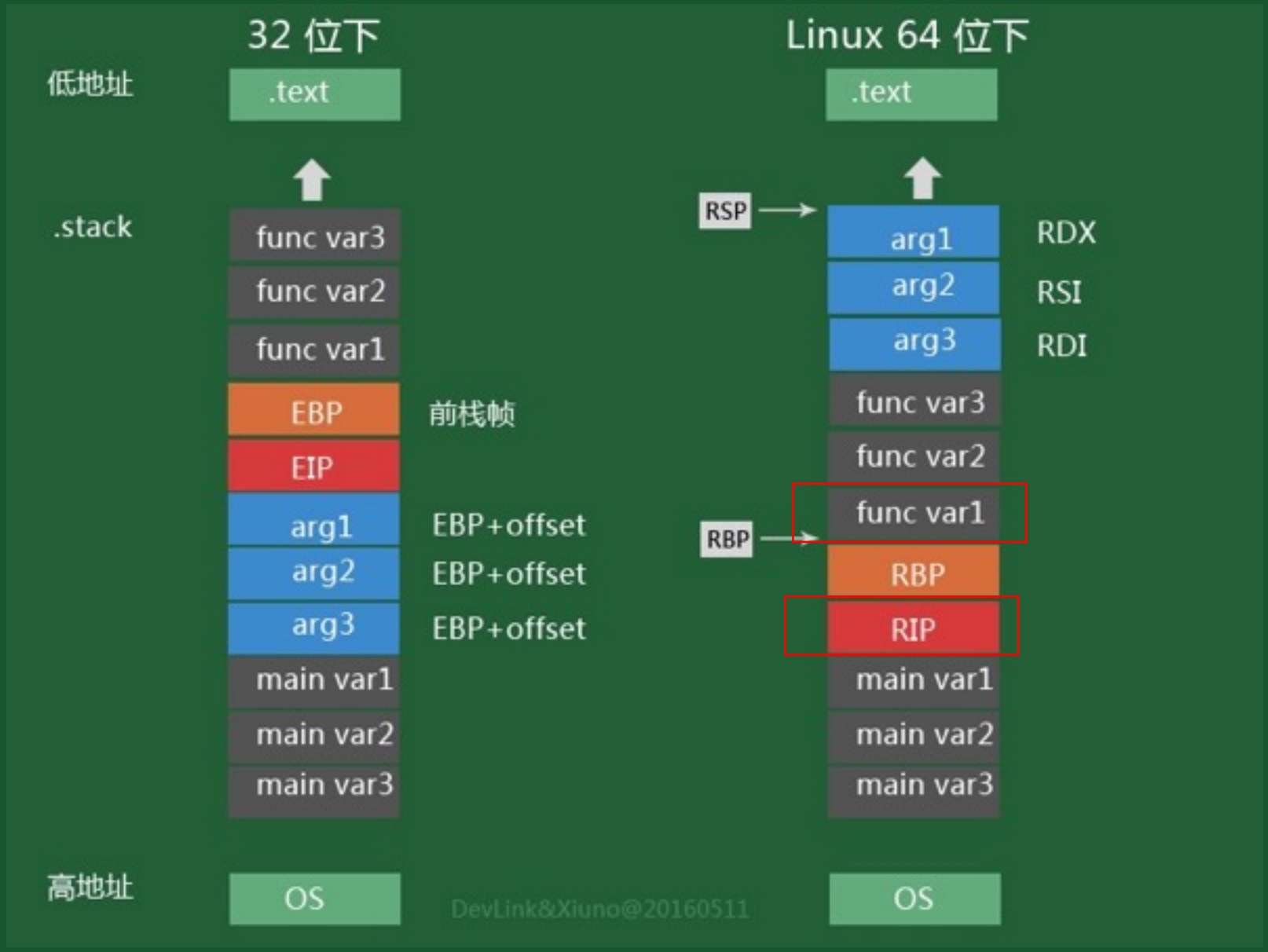
为了贴近实战，我们以 Linux 64 为环境。

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

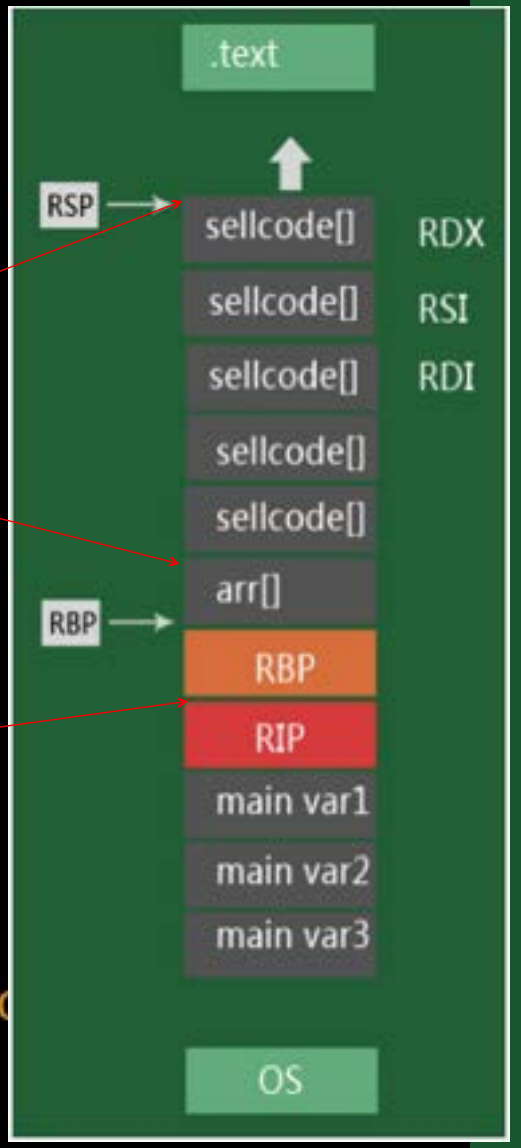


1.4 Linux 64 位下的栈溢出实例程序

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


char **p;

int func() {
        unsigned char arr[] = "ABCDEFG";

        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};

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

        return 0;
}

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

system()

different in proc



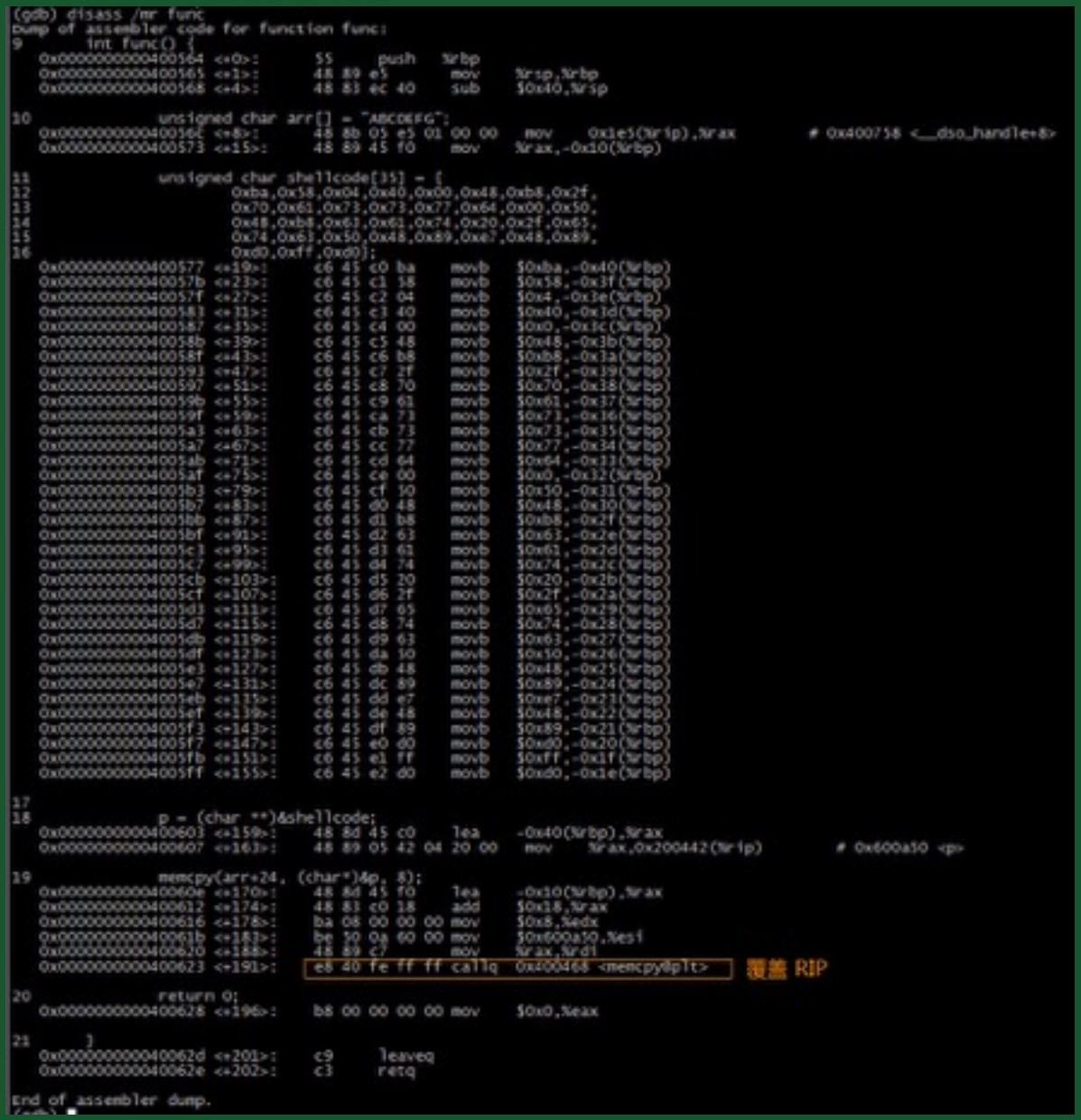

```
gcc -g overflow.c -z execstack -fno-stack-protector -o overflow.o && ./
overflow.o
```

运行结果

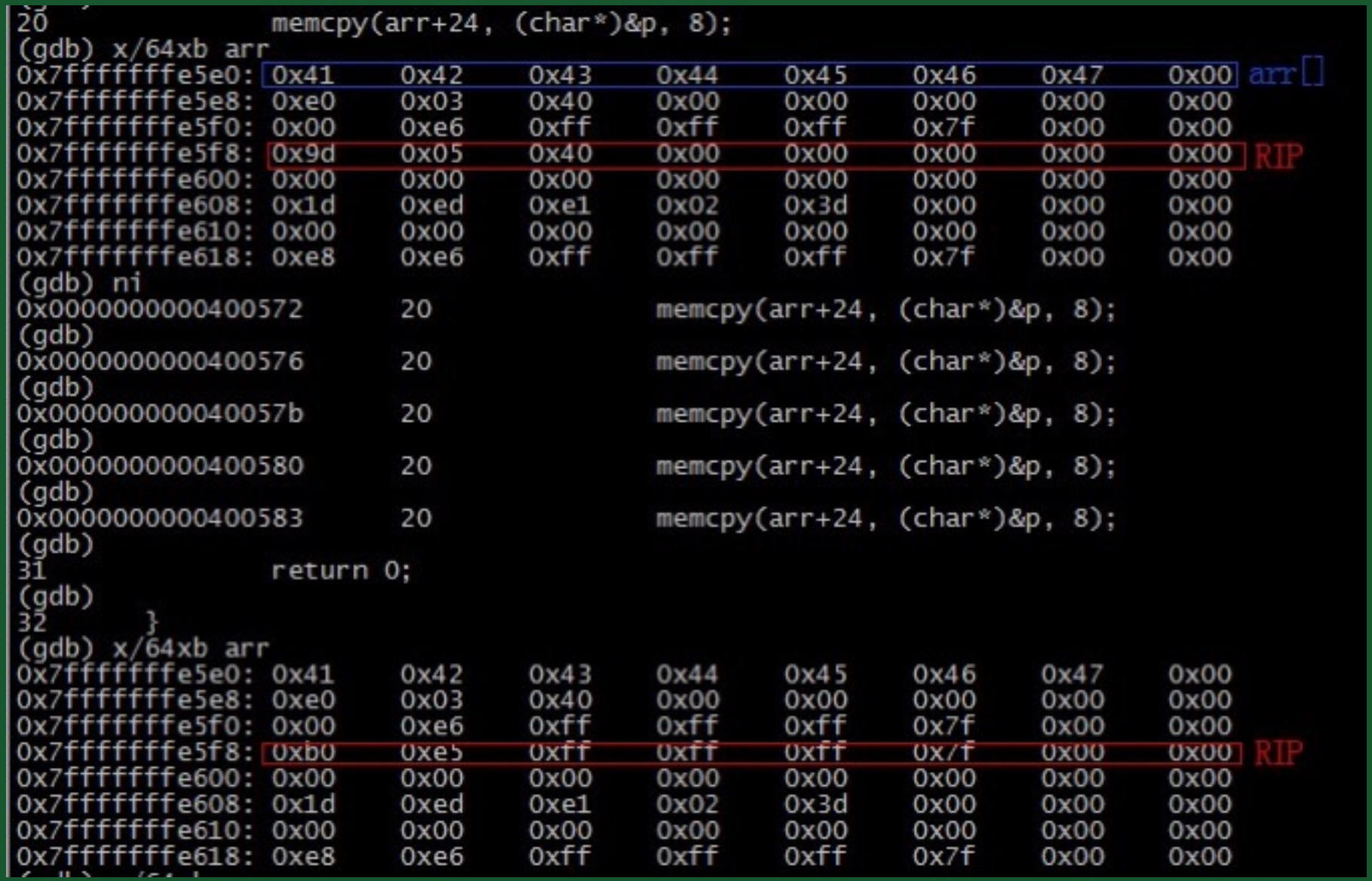
```
[root@xiuno shellcode]# ./overflow.o
system: 400458
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
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
```

shellcode call system(" cat /etc/passwd") successfully

func() 汇编



func() RIP 覆盖


20 memcpy(arr+24, (char*)&p, 8);
(gdb) x/64xb arr
0x7fffffffe5e0: 0x41 0x42 0x43 0x44 0x45 0x46 0x47 0x00 arr[]
0x7fffffffe5e8: 0xe0 0x03 0x40 0x00 0x00 0x00 0x00 0x00
0x7fffffffe5f0: 0x00 0xe6 0xff 0xff 0xff 0x7f 0x00 0x00
0x7fffffffe5f8: 0x9d 0x05 0x40 0x00 0x00 0x00 0x00 0x00 RIP
0x7fffffffe600: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x7fffffffe608: 0x1d 0xed 0xe1 0x02 0x3d 0x00 0x00 0x00
0x7fffffffe610: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x7fffffffe618: 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
0x7fffffffe5e0: 0x41 0x42 0x43 0x44 0x45 0x46 0x47 0x00
0x7fffffffe5e8: 0xe0 0x03 0x40 0x00 0x00 0x00 0x00 0x00
0x7fffffffe5f0: 0x00 0xe6 0xff 0xff 0xff 0x7f 0x00 0x00
0x7fffffffe5f8: 0xb0 0xe5 0xff 0xff 0xff 0x7f 0x00 0x00 RIP
0x7fffffffe600: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x7fffffffe608: 0x1d 0xed 0xe1 0x02 0x3d 0x00 0x00 0x00
0x7fffffffe610: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x7fffffffe618: 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
0x0000000000400612      19              memcpy(arr+24, (char*)&p, 8);
(gdb) ni
0x0000000000400616      19              memcpy(arr+24, (char*)&p, 8);
(gdb)
0x000000000040061b      19              memcpy(arr+24, (char*)&p, 8);
(gdb) ni
0x0000000000400620      19              memcpy(arr+24, (char*)&p, 8);
(gdb)
0x0000000000400623      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
0x000000000040062e in func () at overflow.c:21
21      }
(gdb) si
0x00007fffffffe5a0 in ?? ()
(gdb) x/16i $rip
=> 0x7fffffffe5a0:      mov    $0x400458,%edx
   0x7fffffffe5a5:      movabs $0x6477737361702f,%rax
   0x7fffffffe5af:      push   %rax
   0x7fffffffe5b0:      movabs $0x6374652f20746163,%rax
   0x7fffffffe5ba:      push   %rax
   0x7fffffffe5bb:      mov    %rsp,%rdi
   0x7fffffffe5be:      mov    %rdx,%rax
   0x7fffffffe5c1:      callq  *%rax
   0x7fffffffe5c3:      add    %al,(%rax)
   0x7fffffffe5c5:      add    %al,(%rax)
   0x7fffffffe5c7:      add    %al,(%rax)
   0x7fffffffe5c9:      repz sbb %al,(%rbx)
   0x7fffffffe5cc:      cmp    $0x41000000,%eax
   0x7fffffffe5d1:      rex.X
   0x7fffffffe5d2:      rex.XB
   0x7fffffffe5d3:      rex.R
(gdb)
```

进入了栈区

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
7f425dabe000-7f425dcbe000 ---p 00017000 fc:01 3831
7f425dcbe000-7f425dcbf000 r--p 00017000 fc:01 3831
7f425dcbf000-7f425dcc0000 rw-p 00018000 fc:01 3831
7f425dcc0000-7f425dcc4000 rw-p 00000000 00:00 0
7f425dcc4000-7f425dcc7000 r-xp 00000000 fc:01 4992
7f425dcc7000-7f425dec6000 ---p 00003000 fc:01 4992
7f425dec6000-7f425dec7000 r--p 00002000 fc:01 4992
7f425dec7000-7f425dec8000 rw-p 00003000 fc:01 4992
7f425dec8000-7f425df3a000 r-xp 00000000 fc:01 5219
7f425df3a000-7f425e139000 ---p 00072000 fc:01 5219
7f425e139000-7f425e13a000 r--p 00071000 fc:01 5219
7f425e13a000-7f425e13d000 rw-p 00072000 fc:01 5219
7f425e13d000-7f425e13f000 r-xp 00000000 fc:01 3789
7f425e13f000-7f425e33e000 ---p 00002000 fc:01 3789
7f425e33e000-7f425e33f000 r--p 00001000 fc:01 3789
7f425e33f000-7f425e340000 rw-p 00002000 fc:01 3789
7f425e340000-7f425e356000 r-xp 00000000 fc:01 3833
7f425e356000-7f425e556000 ---p 00016000 fc:01 3833
7f425e556000-7f425e557000 r--p 00016000 fc:01 3833
7f425e557000-7f425e558000 rw-p 00017000 fc:01 3833
7f425e558000-7f425e55a000 rw-p 00000000 00:00 0
7f425e55a000-7f425e6e4000 r-xp 00000000 fc:01 3807
7f425e6e4000-7f425e8e4000 ---p 0018a000 fc:01 3807
7f425e8e4000-7f425e8e8000 r--p 0018a000 fc:01 3807
7f425e8e8000-7f425e8e9000 rw-p 0018e000 fc:01 3807
7f425e8e9000-7f425e8ee000 rw-p 00000000 00:00 0
7f425e8ee000-7f425e904000 r-xp 00000000 fc:01 8205
7f425e904000-7f425eb03000 ---p 00016000 fc:01 8205
7f425eb03000-7f425eb04000 rw-p 00015000 fc:01 8205
7f425eb04000-7f425ebe7000 r-xp 00000000 fc:01 29026
```

```
7f481eefe000-7f481ef15000 r-xp 00000000 fc:01 3831    /lib64/libpthread-2.12.so
7f481ef15000-7f481f115000 ---p 00017000 fc:01 3831    /lib64/libpthread-2.12.so
7f481f115000-7f481f116000 r--p 00017000 fc:01 3831    /lib64/libpthread-2.12.so
7f481f116000-7f481f117000 rw-p 00018000 fc:01 3831    /lib64/libpthread-2.12.so
7f481f117000-7f481f11b000 rw-p 00000000 00:00 0
7f481f11b000-7f481f11e000 r-xp 00000000 fc:01 4992    /lib64/libgpg-error.so.0.5.0
7f481f11e000-7f481f31d000 ---p 00003000 fc:01 4992    /lib64/libgpg-error.so.0.5.0
7f481f31d000-7f481f31e000 r--p 00002000 fc:01 4992    /lib64/libgpg-error.so.0.5.0
7f481f31e000-7f481f31f000 rw-p 00003000 fc:01 4992    /lib64/libgpg-error.so.0.5.0
7f481f31f000-7f481f391000 r-xp 00000000 fc:01 5219    /lib64/libgcrypt.so.11.5.3
7f481f391000-7f481f590000 ---p 00072000 fc:01 5219    /lib64/libgcrypt.so.11.5.3
7f481f590000-7f481f591000 r--p 00071000 fc:01 5219    /lib64/libgcrypt.so.11.5.3
7f481f591000-7f481f594000 rw-p 00072000 fc:01 5219    /lib64/libgcrypt.so.11.5.3
7f481f594000-7f481f596000 r-xp 00000000 fc:01 3789    /lib64/libfreebl3.so
7f481f596000-7f481f795000 ---p 00002000 fc:01 3789    /lib64/libfreebl3.so
7f481f795000-7f481f796000 r--p 00001000 fc:01 3789    /lib64/libfreebl3.so
7f481f796000-7f481f797000 rw-p 00002000 fc:01 3789    /lib64/libfreebl3.so
7f481f797000-7f481f7ad000 r-xp 00000000 fc:01 3833    /lib64/libresolv-2.12.so
7f481f7ad000-7f481f9ad000 ---p 00016000 fc:01 3833    /lib64/libresolv-2.12.so
7f481f9ad000-7f481f9ae000 r--p 00016000 fc:01 3833    /lib64/libresolv-2.12.so
7f481f9ae000-7f481f9af000 rw-p 00017000 fc:01 3833    /lib64/libresolv-2.12.so
7f481f9af000-7f481f9b1000 rw-p 00000000 00:00 0
7f481f9b1000-7f481fb3b000 r-xp 00000000 fc:01 3807    /lib64/libc-2.12.so
7f481fb3b000-7f481fd3b000 ---p 0018a000 fc:01 3807    /lib64/libc-2.12.so
7f481fd3b000-7f481fd3f000 r--p 0018a000 fc:01 3807    /lib64/libc-2.12.so
7f481fd3f000-7f481fd40000 rw-p 0018e000 fc:01 3807    /lib64/libc-2.12.so
7f481fd40000-7f481fd45000 rw-p 00000000 00:00 0
7f481fd45000-7f481fd5b000 r-xp 00000000 fc:01 8205    /lib64/libgcc_s-4.4.7-20120601.so
7f481fd5b000-7f481ff5a000 ---p 00016000 fc:01 8205    /lib64/libgcc_s-4.4.7-20120601.so
7f481ff5a000-7f481ff5b000 rw-p 00015000 fc:01 8205    /lib64/libgcc_s-4.4.7-20120601.so
7f481ff5b000-7f4820003e000 r-xp 00000000 fc:01 29026  /usr/local/lib/libiconv.so.2.5.1
```

Linux 64 位下的 shellcode 提取

```php
<?php
echo bin2hex('cat /etc/passwd');

// 636174202f6574632f706173737764

// 整理字符串:
// 63 61 74 20 2f 65 74 63 2f 70 61 73 73 77 64 00
// 63 61 74 20 2f 65 74 63
// 2f 70 61 73 73 77 64 00

// intel little endian
echo "\r\n";
$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';
echo str_replace(' ', ',0x', $s);
echo "\r\n";
echo str_word_count($s, NULL, ' ');
?>
```

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

prepare to fetch shellcode

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

void func() {
        char *arr;
        asm volatile(
                "mov $0x0064777373617 02f,%%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;
}
```

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(
   0x00000000004004c8 <+4>:     ba d0 03 40 00  mov    $0x4003d0,%edx
   0x00000000004004cd <+9>:     48 b8 2f 70 61 73 73 77 64 00   movabs $0x647773736170 2f,%rax
   0x00000000004004d7 <+19>:    50      push   %rax
   0x00000000004004d8 <+20>:    48 b8 63 61 74 20 2f 65 74 63   movabs $0x6374652f20746163,%rax
   0x00000000004004e2 <+30>:    50      push   %rax
   0x00000000004004e3 <+31>:    48 89 e7        mov    %rsp,%rdi
   0x00000000004004e6 <+34>:    48 89 d0        mov    %rdx,%rax
   0x00000000004004e9 <+37>:    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.
```

system()

shellcode

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?

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。

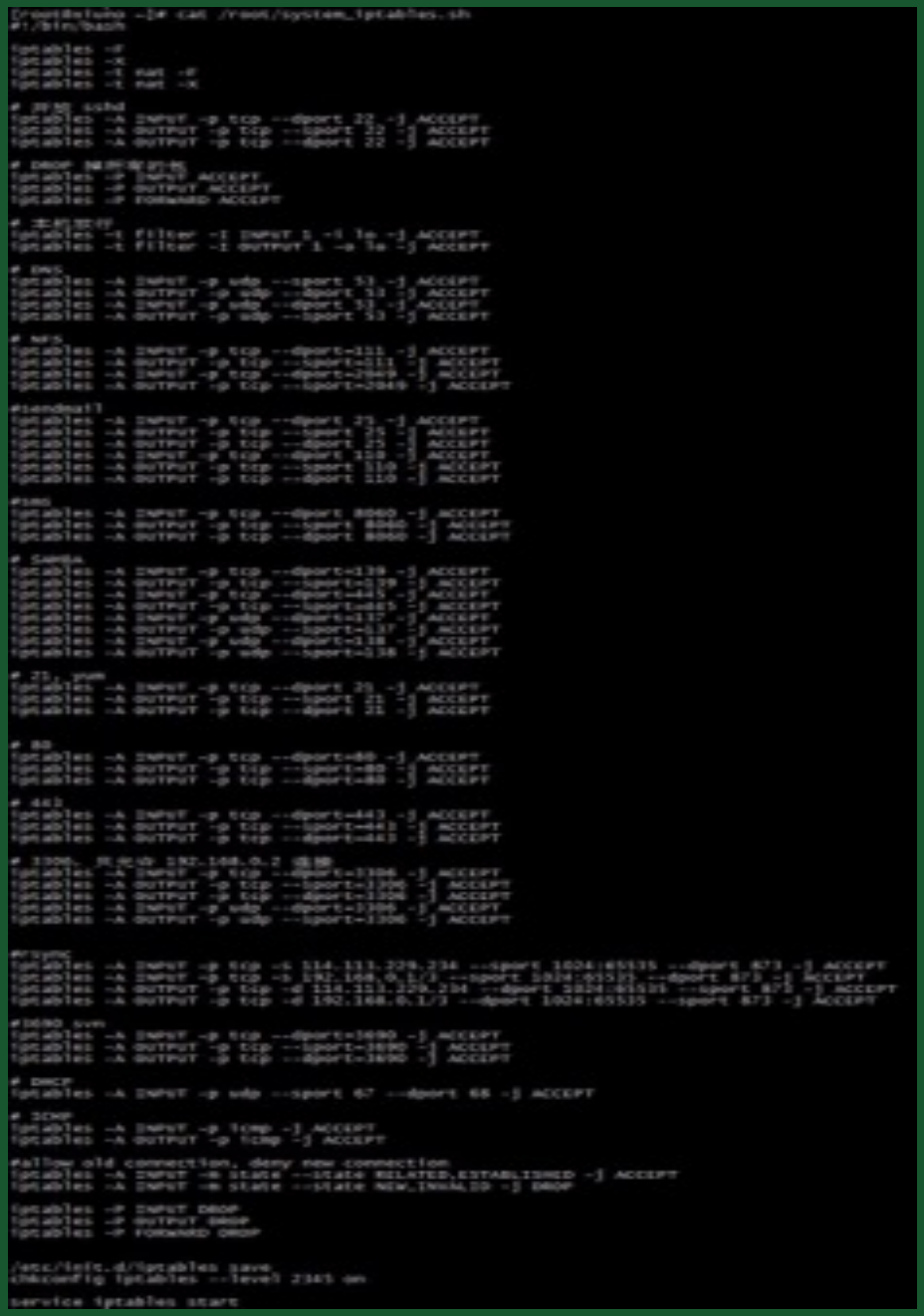
关闭不必要的服务

```
[root@xiuno ~]# cat /root/system_optimzer.sh
#!/bin/bash
SERVICES="abrt-ccpp abrt-oops abrtd 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 pcscd portmap portreserve postfix psacct rpcgs
sd quota_nld rdisc restorecond rpcidmapd xfs sendmail saslauthd smartd sssd wdae
mon wpa_supplicant ypbind  yum-updatesd"
for service in $SERVICES
do
        chkconfig $service off
        service $service stop
done
```

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@20151111822513 ~]# 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@20151111822513 ~]#
```

1.6 本机严格的防火墙策略



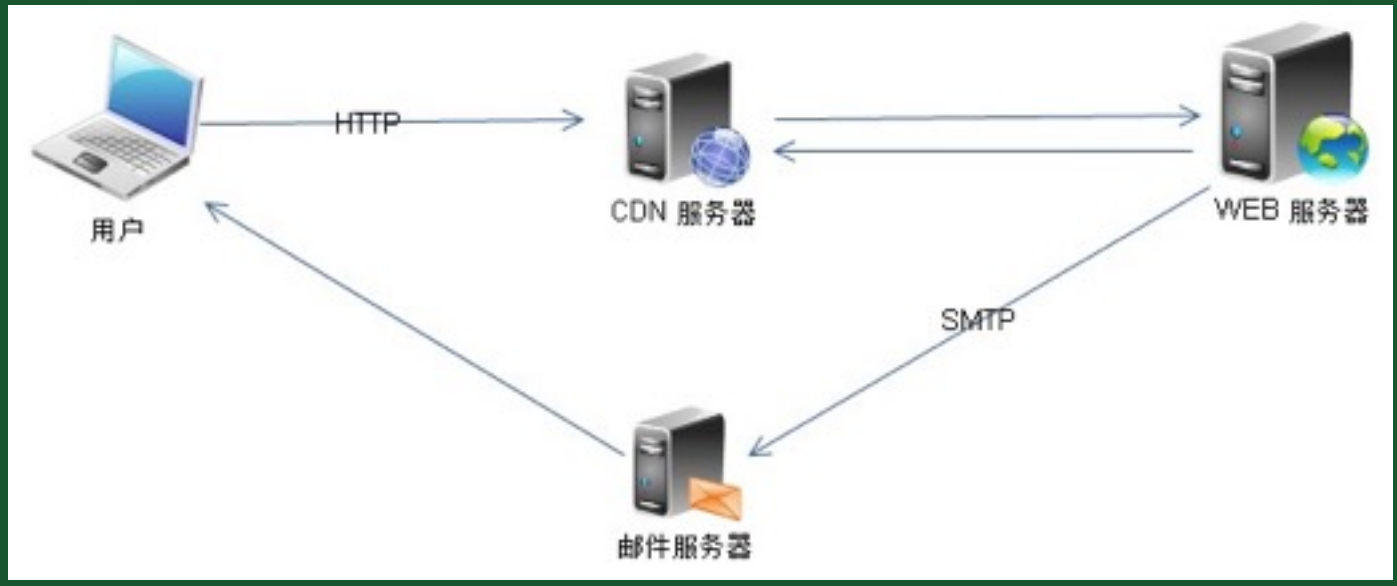
有实力可以考虑硬件防火墙-价格比较昂贵

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


Google
in:spam
Gmail
永久删除
这不是垃圾邮件
更多操作
Xiuno 轻论坛 <axiuno@sina.com>
为什么此邮件会在"垃圾邮件"中？因为它类似于我们的垃圾邮件过滤器检测到的邮件。 了解详情
重设密码验证码 【Xiuno 轻论坛】
10:46 (3分钟前)
回复
转发
过滤此类邮件
打印
删除此邮件
屏蔽"Xiuno 轻论坛"
举报网上诱骗
显示原始邮件
翻译邮件
标记为未读
Xiuno 轻论坛
添加到圈子

隐藏 IP - CDN

```
Delivered-To: axiuno@gmail.com
Received: by 10.129.161.196 with SMTP id y187csp469436ywg;
        Wed, 11 May 2016 19:46:26 -0700 (PDT)
X-Received: by 10.157.24.49 with SMTP id b46mr1055173ote.93.1463021186708;
        Wed, 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])
        by mx.google.com with SMTP id qk8si371100oeb.26.2016.05.11.19.46.25
        for <axiuno@gmail.com>;
        Wed, 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)
X-Sender: axiuno@sina.com
X-Auth-ID: axiuno@sina.com
X-SMAIL-MID: 2009525767625
Date: Thu, 12 May 2016 10:46:23 +0800
Return-Path: axiuno@sina.com
To: =?UTF-8?B?WG11bm8g6L276K665Z2b?= <axiuno@gmail.com>
From: =?UTF-8?B?WG11bm8g6L276K665Z2b?= <axiuno@sina.com>
Reply-To: =?UTF-8?B?WG11bm8g6L276K665Z2b?= <axiuno@sina.com>
Subject: =?UTF-8?B?6YeN6K6+5a+G56CB6aqM6K+B56CB77yaMjkxMDc0IC0g44CQWG11bm8g?=
 =?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"
```

received by gmail server

send from sina mail server

leak the real web server IP

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

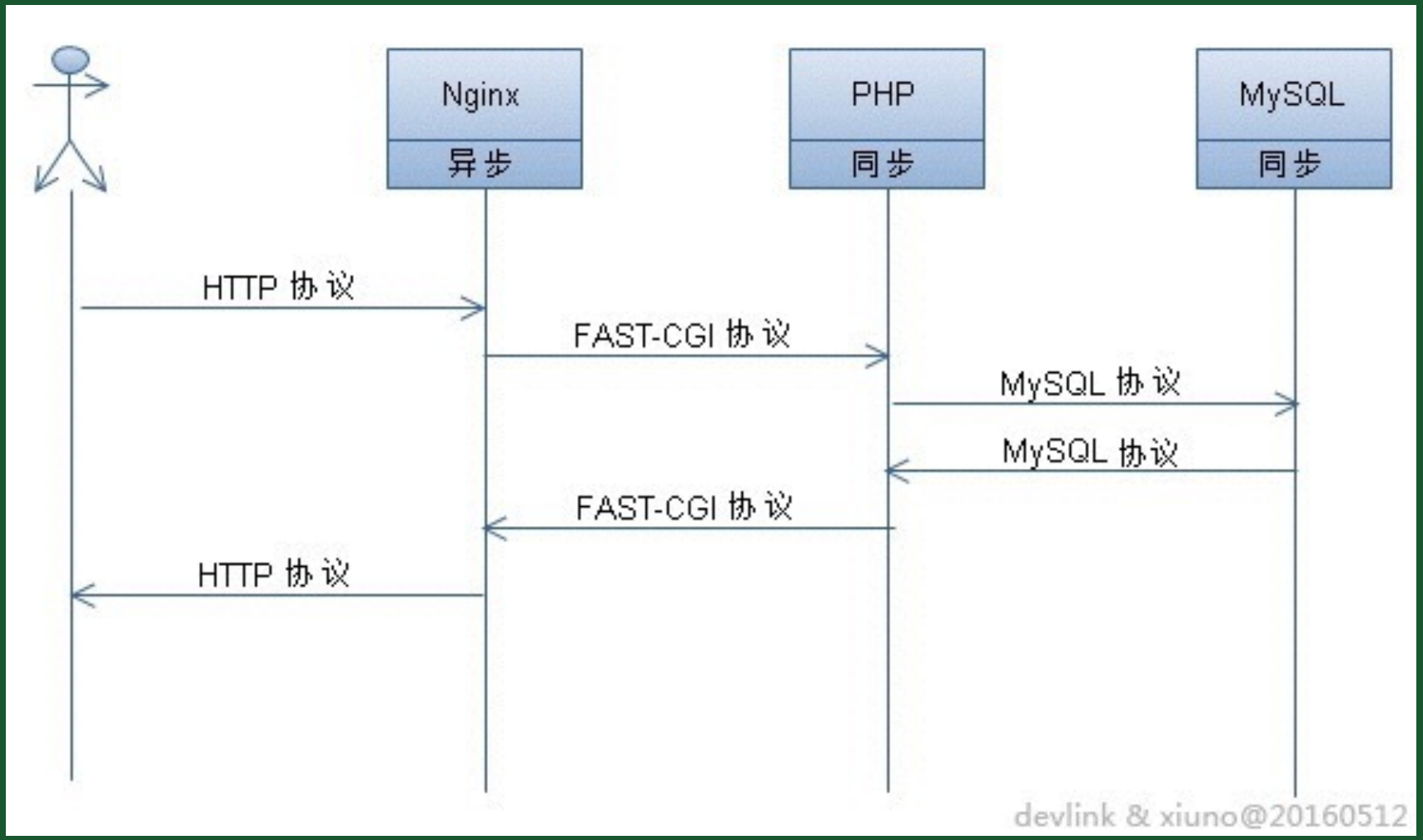
- 攻击 CDN 服务器（理论上，实际难度比较大）
- IP 段扫描，对比正文，找出所有 CDN 和 真实主机 IP

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	多线程 + 同步
nodejs, nginx, lighttpd, redis, 业务相对简单的网游：如 QQ 棋牌	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：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。	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
<?php
$uid = $_GET['uid'];
$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);

?>
```

可被外部控制

现在应该没人这么写了吧!

老生常谈：SQL 注射成因

```php
<?php
$username = $_GET['username'];
$username = addslashes($username);
$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);

?>
```

安全转义

最起码也得这样！

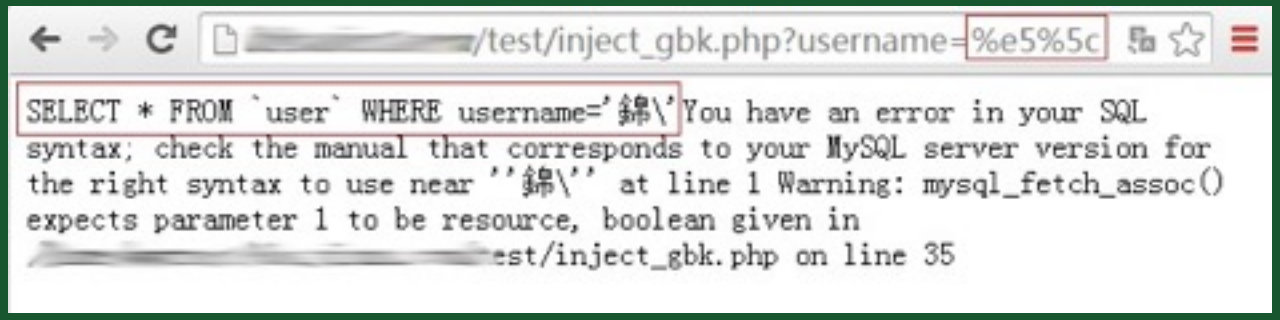
2.5 SQL 注射 GBK 漏洞

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
<?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\'



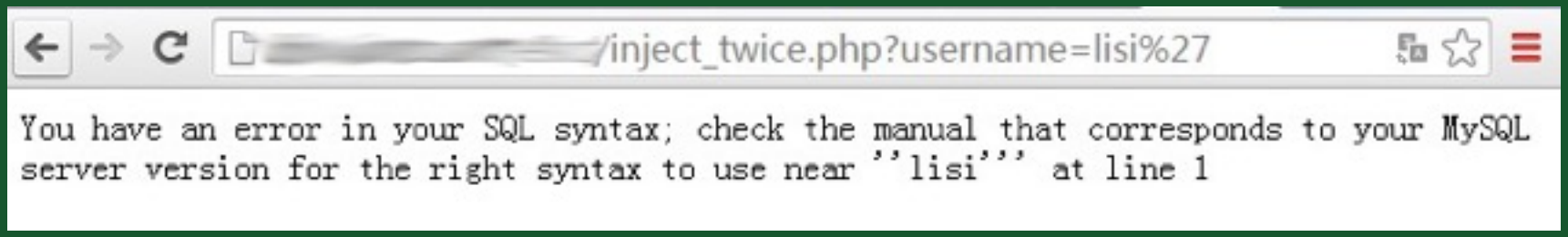

```
mysql> select version();
+-----------+
| version() |
+-----------+
| 5.5.37    |
+-----------+
1 row in set (0.03 sec)
```

2.6 二次注射

```php
<?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"



2.7 不安全的单字节操作函数，如： substr()

```php
<?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='0123456789012345678901234567891\

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_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_reaplce() 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
<?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!');
}

move_uploaded_file($tmpfile, './upload/'.$filename);

?>
```

应该采用白名单，严格过滤 $filename

写出这种代码基本就 GAME OVER 了，各种死法。

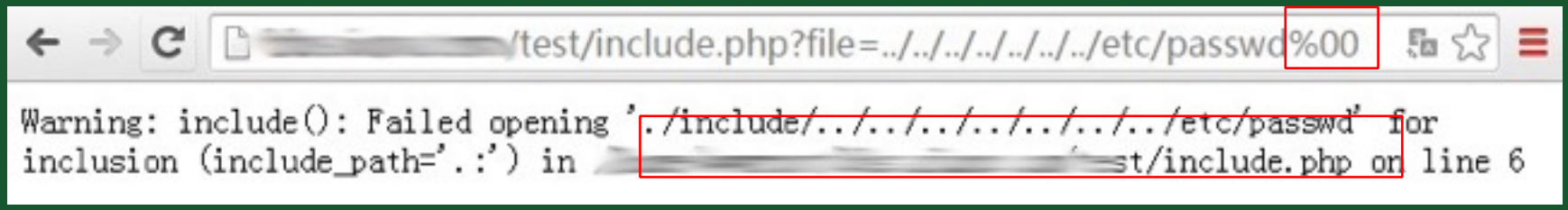
123.php%00.jpg

123.asa IIS 默认支持的后缀

../../static/common.js

2.9 危险的 include $var;

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

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



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。

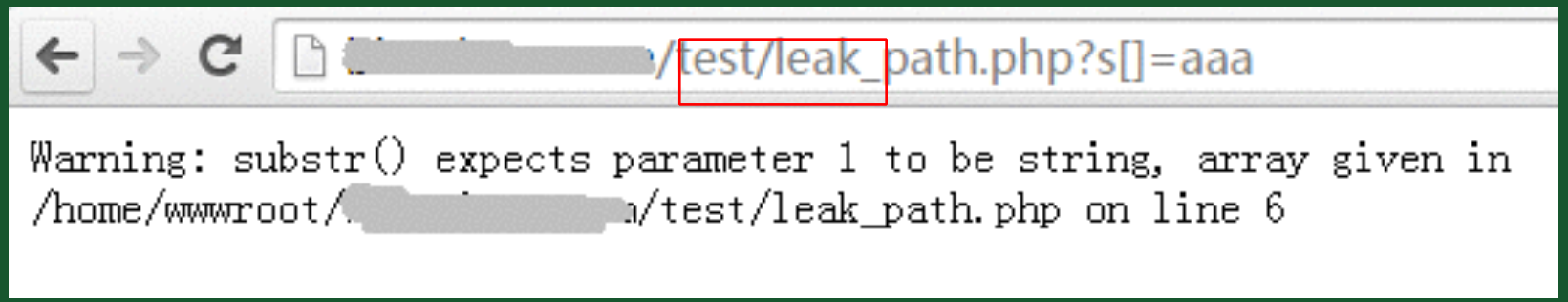
```
$ip = $_SERVER['HTTP_X_FORWARDED_FOR'];

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
<?php

error_reporting(E_ALL);

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

echo $s;

?>
```



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：

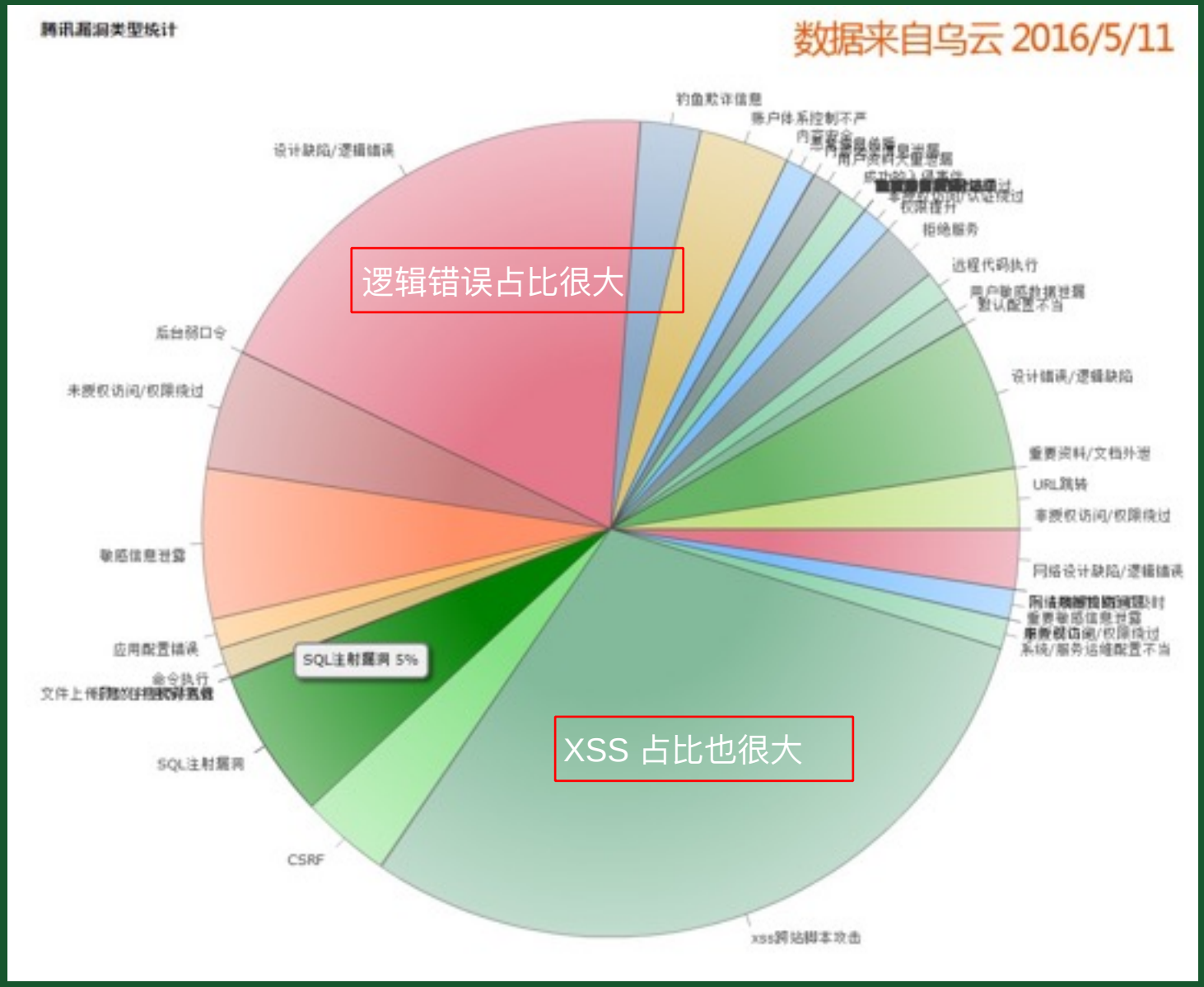
```php
<?php
$a = 'a';
if($a > 0) {
        echo 'Ok';
} else {
        echo 'error';
}
?>
```

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
钓鱼欺诈信息
设计缺陷/逻辑错误
逻辑错误占比很大
后台弱口令
未授权访问/权限绕过
敏感信息泄露
应用配置错误
命令执行
SQL注射漏洞 5%
SQL注射漏洞
CSRF
XSS 占比也很大
拒绝服务
远程代码执行
设计错误/逻辑缺陷
重要资料/文档外泄
URL跳转
未授权访问/权限绕过
网络设计缺陷/逻辑错误
系统/服务运维配置不当

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：

```
if($operateId == 1){
    $date = date("Ymd");
    $dest = $CONFIG->basePath."data/files/".$date."/";
    $COMMON->createDir($dest);
    //if (!is_dir($dest))    mkdir($dest, 0777);
    $nameExt = strtolower($COMMON->getFileExtName($_FILES['Filedata']['name']));
    $allowedType = array('jpg', 'gif', 'bmp', 'png', 'jpeg');
    if(!in_array($nameExt, $allowedType)){
        $msg = 0;
    }
    if(empty($msg)){
        $filename = getmicrotime().'.'.$nameExt;
        $file_url = urlencode($CONFIG->baseUrl.'data/files/'.$date."/".$filename);
        $filename = $dest.$filename;
        if(empty($_FILES['Filedata']['error'])){
            move_uploaded_file($_FILES['Filedata']['tmp_name'],$filename);
        }
        if (file_exists($filename)){
            //$msg = 1;
            $msg = $file_url;
            @chmod($filename, 0444);
        }else{
            $msg = 0;
        }
    }
    $outMsg = "fileUrl=".$msg;
    $_SESSION["eoutmsg"] = $outMsg;
    exit;
}
```

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 银联

http://**.**.**.**/5107/upload/uploadFlash.php 迅雷

http://**.**.**.**/5107/upload/uploadFlash.php 太平洋保险

http://**.**.**.**/5107/upload/uploadFlash.php 人寿保险

http://**.**.**.**/5107/upload/uploadFlash.php 海南航空

http://**.**.**.**/5107/upload/uploadFlash.php 网维大师

http://**.**.**.**/5107/upload/uploadFlash.php 海尔电器

http://**.**.**.**/5107/upload/uploadFlash.php 金山毒霸

http://**.**.**.**/5107/upload/uploadFlash.php 盛大在线

http://**.**.**.**/5107/upload/uploadFlash.php 大成基金

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。

<img src=”<?php echo $user['avatar_url'];?>”>

如果用户输入 1.jpg“ onload=”alert(123)

那么结果就成了：

<img src=”1.jpg”onload=”alert(123)” >

```
$avatar_url = htmlspecialchars($_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 语法解析的类库：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

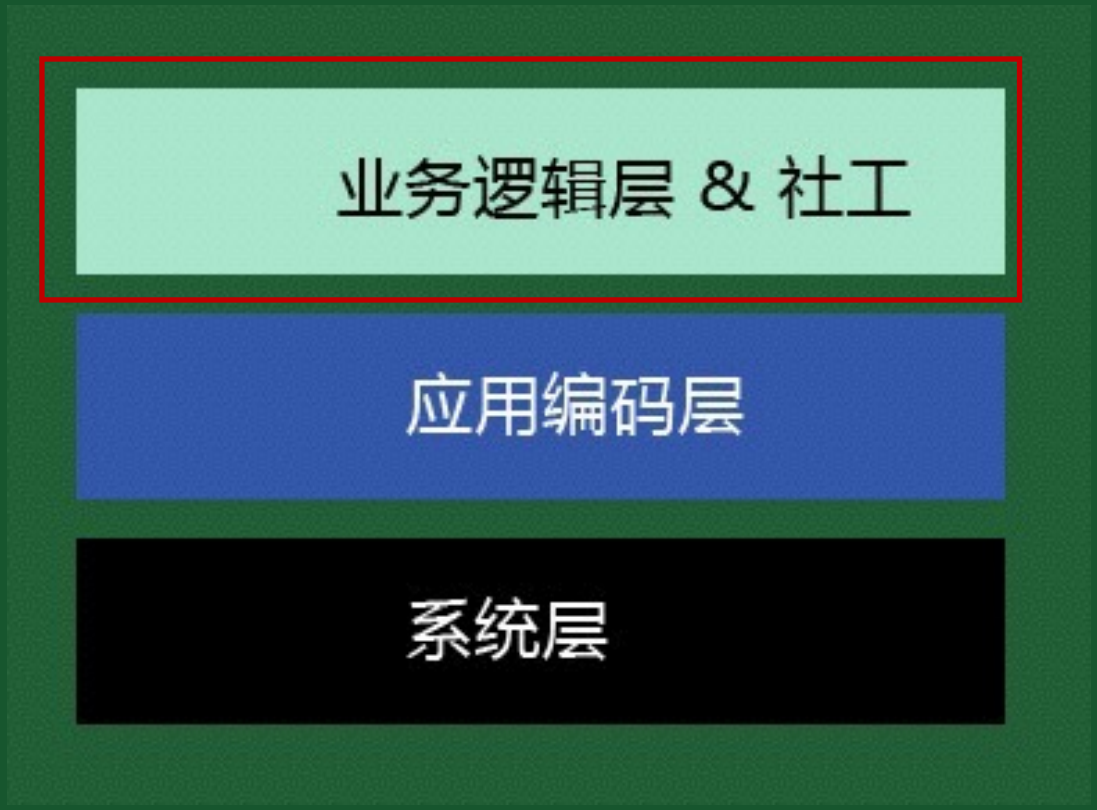
HTML CSS 白名单设置

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

```php
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。

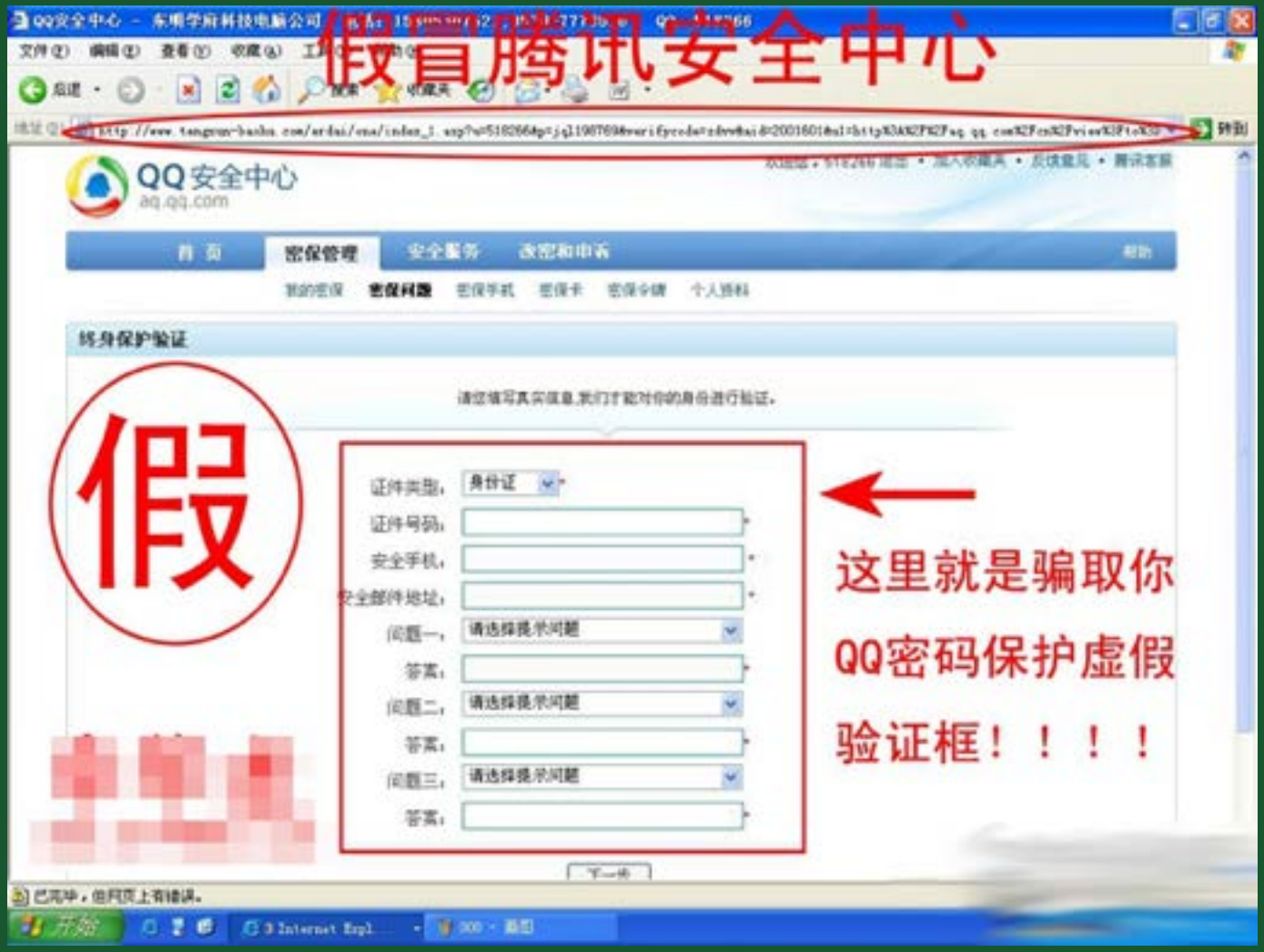
3.3 密码排列组合

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