华中科技大学
HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

网络空间安全学院

整数溢出漏洞

网络空间安全学院 慕冬亮
Email : dzm91@hust.edu.cn

[1] Basic Integer Overflows

提纲

1. 整数溢出背景
2. 什么是整数溢出?
3. 整数溢出典型问题
4. 整数溢出真实案例
5. 如何防范整数溢出?
6. 其它溢出

1. 整数溢出背景

CVE-2016-5636 : **Integer overflow** in the get_data function in ...
www.cvedetails.com › cve › CVE-2016-5636
CVE-2016-5636 : **Integer overflow** in the get_data function in zipimport.c in CPython (aka

CVE-2017-2888 : An exploitable **integer overflow** vulnerability exists ...
www.cvedetails.com › cve › CVE-2017-2888
Feb 11, 2021 ... CVE-2017-2888 : An exploitable **integer overflow** vulnerability exists whe

CVE-2021-20203 : An **integer overflow** issue was found in the ...
www.cvedetails.com › cve-details
Apr 11, 2021 ... CVE-2021-20203 : An **integer overflow** issue was found in the vmxnet3 N

CVE-2021-29338 : **Integer Overflow** in OpenJPEG v2.4.0 allows ...
www.cvedetails.com › cve-details
Jun 12, 2021 ... CVE-2021-29338 : **Integer Overflow** in OpenJPEG v2.4.0 allows remote





[1] https://www.cvedetails.com/google-search-results.php?q=integer+overflow&sa=Search

1. 整数溢出背景

2010-05-24 多家厂商 rpc.pcnfsd 服务整数溢出漏洞

NSFOUCS ID: 15091

综述: rpc.pcnfsd 是一个在网络上提供认证和打印服务的 RPC 守护进程，运行在大量 Unix 类操作系统上。多个厂商的 Unix 系统中所使用的 rpc.pcnfsd 服务在处理 RPC 请求时存在整数溢出漏洞。

危害: 远程攻击者可以通过发送特制的 rpc 请求来触发此漏洞，从而控制服务器系统。

2010-05-12 Outlook Express 和 Windows Mail STAT 响应整数溢出漏洞（MS10-030）

NSFOUCS ID: 15002

综述: Outlook Express 和 Windows Mail 都是 Windows 操作系统中默认捆绑的邮件和新闻组客户端。Outlook Express 和 Windows Mail 客户端所使用的通用库验证特制邮件响应的方式存在整数溢出漏洞。如果用户受骗使用 POP3 和 IMAP 邮件协议连接到了恶意的服务器并收到了畸形的 STAT 响应就会触发这个溢出，可能导致在用户系统上执行任意代码。

危害: 远程攻击者可在用户系统上执行任意代码。

2013-11-07 Google Android 签名验证安全措施绕过漏洞(含整数溢出原因)

NSFOUCS ID: 25224

综述: Android 是基于 Linux 开放性内核的操作系统，是 Google 公司在 2007 年 11 月 5 日公布的手机操作系统。Android 4.4 及其他版本存在安全限制绕过漏洞，攻击者可利用此漏洞绕过某些安全限制以执行未授权操作。

危害: 攻击者可以利用此漏洞绕过安卓签名检查，从而控制受害者系统。

[1] http://www.nsfocus.net/index.php?act=sec_bug

1. 整数溢出背景

Rank	ID	Name	Score	2020 Rank Change
[1]	CWE-787	Out-of-bounds Write	65.93	+1
[2]	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	46.84	-1
[3]	CWE-125	Out-of-bounds Read	24.9	+1
[4]	CWE-20	Improper Input Validation	20.47	-1
[5]	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	19.55	+5
[6]	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	19.54	0
[7]	CWE-416	Use After Free	16.83	+1
[8]	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	14.69	+4
[9]	CWE-352	Cross-Site Request Forgery (CSRF)	14.46	0
[10]	CWE-434	Unrestricted Upload of File with Dangerous Type	8.45	+5
[11]	CWE-306	Missing Authentication for Critical Function	7.93	+13
[12]	CWE-190	Integer Overflow or Wraparound	7.12	-1
[13]	CWE-502	Deserialization of Untrusted Data	6.71	+8
[14]	CWE-287	Improper Authentication	6.58	0
[15]	CWE-476	NULL Pointer Dereference	6.54	-2
[16]	CWE-798	Use of Hard-coded Credentials	6.27	+4
[17]	CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer	5.84	-12
[18]	CWE-862	Missing Authorization	5.47	+7
[19]	CWE-276	Incorrect Default Permissions	5.09	+22
[20]	CWE-200	Exposure of Sensitive Information to an Unauthorized Actor	4.74	-13
[21]	CWE-522	Insufficiently Protected Credentials	4.21	-3
[22]	CWE-732	Incorrect Permission Assignment for Critical Resource	4.2	-6
[23]	CWE-611	Improper Restriction of XML External Entity Reference	4.02	-4
[24]	CWE-918	Server-Side Request Forgery (SSRF)	3.78	+3
[25]	CWE-77	Improper Neutralization of Special Elements used in a Command ('Command Injection')	3.58	+6

2. 什么是整数溢出？

- 计算机中整数都有一个宽度（例如Win7下VC6编译器中int类型为32位）

- 当试图保存一个比它可以表示的最大值还大的数时，就会发生整数溢出

- 整数溢出将导致“不确定性行为”。比如完全忽略该溢出或终止进程。

大多数编译器都会忽略这种溢出，这可能会导致不确定或错误的值保存在了整数变量中

2. 什么是整数溢出？

整数值范围

编译器类型	数据类型	数据名称	最小值	最大值
VC++ 6.0	unsigned short	无符号短整型	0	65535
	short	短整型	-32,768	32,767
	unsigned int	无符号整型	0	4,294,967,295
	int	整型	-2,147,483,648	2,147,483,647
	unsigned long	无符号长整型	0	4,294,967,295
	long	长整型	-2,147,483,648	2,147,483,647
	unsigned __int64	无符号64位整数	0	18,446,744,073,709,500,000
	__int64	64位整数	-9,223,372,036,854,770,000	9,223,372,036,854,770,000
VS2012 C#	ushort	无符号短整型	0	65535
	short	短整型	-32,768	32,767
	uint	无符号整型	0	4,294,967,295
	int	整型	-2,147,483,648	2,147,483,647
	ulong	无符号长整型	0	18,446,744,073,709,500,000
	long	长整型	-9,223,372,036,854,770,000	9,223,372,036,854,770,000

大多数编译器都会忽略这种溢出，这可能会导致不确定或错误的值保存在了整数变量中

2. 什么是整数溢出？

- 2个无符号的整数, a和b， 2个数都是32位字节长

a = 0xFFFFFFFFU ← U声明0xFFFFFFFF为无符号数
b = 0x1U
r = a + b = ?

r = (0xFFFFFFFFU + 0x1U) % 0x100000000
r = (0x100000000) % 0x100000000
r = 0

“环绕”与“截断”处理

2. 什么是整数溢出？

- 3个有符号的整数a，b，r， 3个数都是32位长

```
#include <stdio.h>
#include <stdint.h>

int main()
{
        int32_t a = 0x7fffffff, b = 1, r;
        r = a + b;
        printf("a = %d\n", a);
        printf("b = %d\n", b);
        printf("r = %d\n", r);
}
```

a = 0x7fffffff
b = 1
r = a + b
r = ?

r = (0x7fffffff + 0x1) = 0x80000000
r = -2147483648

2. 什么是整数溢出？

ISO C99:

“A computation involving unsigned operands can never overflow, because a result that cannot be represented by the resulting unsigned integer type is reduced modulo the number that is one greater than the largest value that can be represented by the resulting type.”

涉及到无符号操作数计算的时候从不会溢出, 因为结果不能被无符号类型表示的时候, 就会对比该类型能表示的最大值还大的数求余. 这样就能用该结果来表示这种类型了.

3. 整数溢出典型问题(1)

示例代码1

(1)请指出是否存在问题？(2)原因？(3)你还有更隐蔽的实例代码吗？

```
void main(int argc, char* argv[])
{
    unsigned short s;
    int i;
    char buf[80];
    i = atoi(argv[1]);
    s = i;                          [1]


}
```

3. 整数溢出典型问题(2)

示例代码2

(1)请指出是否存在问题？(2) 原因？(3) 你还有更好实例代码吗?

```
void copy_int_array(int *array, int len)
{
    int *myarray, i;
    myarray = malloc(len * sizeof(int));  [1]
    if (myarray == NULL) return;



   free(myarray);
}
```

3. 整数溢出典型问题(3)

示例代码3

(1)请指出是否存在问题？(2) 原因？(3) 你还有更好的实例代码吗?

```
int store_data(char *buf, int len)
{
    char data[256];
    if (len > 256)
        return -1;
    return memcpy(data, buf, len);      [1]
}
void *memcpy(void *restrict dest, const void
    *restrict src, size_t n);
```

3. 整数溢出典型问题

(1) 宽度溢出　(2) 运算溢出　(3) 符号溢出

```
void main(int argc, char* argv[])
{
        unsigned short s;
        int i;
        char buf[80];
        i = atoi(argv[1]);
        s = i;


}
```

```
void copy_int_array(int *array, int len)
{
        int *myarray, i;
        myarray=malloc(len*sizeof(int));
        if (myarray == NULL) return;
        free(myarray);

}
```

```
int store_data(char *buf, int len)
{
    char data[256];

    if (len > 256)
        return -1;

    return memcpy(data, buf, len);
}
```

编译器默认运算规律：　unsigned long > long > unsigned int > int > unsigned short > short > unsigned char > char

无符号数转换为更大的数据类型时，需要执行零扩展;
有符号数转换为更大的数据类型时，需要执行符号扩展

3. 整数溢出表现形式(1)

示例代码 1

(1)如何利用其中的整数溢出进行攻击?

```
void main(int argc, char* argv[])
{
    unsigned short s;
    int i;
    char buf[80];
    i = atoi(argv[1]);
    s = i;                              [1]
    if (s >= 80) return;
    memcpy(buf, argv[2], i);
}
```

3. 整数溢出表现形式(2)

示例代码 2

(1)如何利用其中的整数溢出进行攻击?

```
void copy_int_array(int *array, int len)
{
    int *myarray, i;
    myarray = malloc(len * sizeof(int));   [1]
    if (myarray == NULL) return;
   for(i=0; i<len; i++)
          myarray[i] = arrary[i];
   free(myarray);
}
```

3. 整数溢出表现形式(3)

示例代码 3

(1)如何利用其中的整数溢出进行攻击?

```
int store_data(char *buf, int len)
{
    char data[256];
    if (len > 256)
        return -1;
    return memcpy(data, buf, len);   [1]
}
void *memcpy(void *restrict dest, const void
    *restrict src, size_t n);
```

4. 整数溢出真实案例

- PHP CVE-2007-1001

在PHP的libgd库中，函数(1) createwbmp，(2) readwbmp 中存在多个整数溢出。该整数溢出允许攻击者通过具有很大的长或宽的WBMP图片来触发缓冲区溢出，从而执行任意代码。

```
if ((wbmp->bitmap = (int *) safe_emalloc(wbmp->width * wbmp->height,
sizeof(int), 0)) == NULL)
```

```
pos = 0;
for (row = 0; row < wbmp->height; row++)
  {
    for (col = 0; col < wbmp->width;)
```

修复方法：对 wbmap->width / height 进行检查，即

if (wbmp->width > INT_MAX / wbmap->height)

[1] https://github.com/VulnReproduction/LinuxFlaw/tree/master/CVE-2007-1001

4. 整数溢出真实案例

- Python CVE-2016-5636

Python中 get_data 函数存在整数溢出，该溢出会允许攻击者通过一个负值来触发缓冲区溢出。

```
1116     bytes_size = compress == 0 ? data_size : data_size + 1;
1117     if (bytes_size == 0)
1118         bytes_size++;
1119     raw_data = PyBytes_FromStringAndSize((char *)NULL, bytes_size);
```

如果compres变量不等于0，bytes_size 将等于 data_size + 1。程序未对data_size进行验证，当 data_size 为 -1 时，bytes_size 结果为0，导致分配的内存过小，后续会出现缓冲区溢出。

修复方法：对 data_size 进行检查，即

if (data_size > LONG_MAX - 1)

[1] https://github.com/VulnReproduction/LinuxFlaw/tree/master/CVE-2016-5636

4. 整数溢出真实案例

- Linux Kernel CVE-2014-2851

```
251 int ping_init_sock(struct sock *sk)
252 {
253         struct net *net = sock_net(sk);
254         kgid_t group = current_egid();
255         struct group_info *group_info = get_current_groups();
256         int i, j, count = group_info->ngroups;
257         kgid_t low, high;
258
259         inet_get_ping_group_range_net(net, &low, &high);
260         if (gid_lte(low, group) && gid_lte(group, high))
261                 return 0;
262
263         for (i = 0; i < group_info->nblocks; i++) {
264                 int cp_count = min_t(int, NGROUPS_PER_BLOCK, count);
265                 for (j = 0; j < cp_count; j++) {
266                         kgid_t gid = group_info->blocks[i][j];
267                         if (gid_lte(low, gid) && gid_lte(gid, high))
268                                 return 0;
269                 }
270
271                 count -= cp_count;
272         }
273
274         return -EACCES;
275 }
```

```
/*
 * COW Supplementary groups list
 */
struct group_info {
        atomic_t        usage;
        int             ngroups;
        kgid_t          gid[0];
} __randomize_layout;
```

```
typedef struct {
        int counter;
} atomic_t;
```

5. 如何防范整数溢出？

- **整数安全意识**

  形成关于特殊数据输入的意识，比如之前先确定最大和最小输入，使用合适的数据类型。

- **避免隐患运算直接操作**

  尽量避免对两个正数相加或相乘之后，再取结果比较，

  len1+len < MAX_IINFO 应该改成：

  ```
  if  (MAX_INFO  - len1 > len2)
      ……
  ```

- **越界判断**

  在使用变量申请内存，或者作为数组下标时，注意对越界的监测。

- **…代码审计、安全测试**

5. 如何防范整数溢出？

- 从今天开始，强烈建议大家记住整数溢出隐患。

  整数安全意识是最根本的

练习

```
int main()
{
  char Buf[1024];
  short a = (short)0xFFFF;
  int b;
  b = a;
  printf("b=%d    \r\n",b);
  b = (a & 0xFFFF);
  printf("b=%d    \r\n",b);
  return 1;
}
```

华中科技大学

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

网络空间安全学院

Thank You !