缓冲区溢出攻击

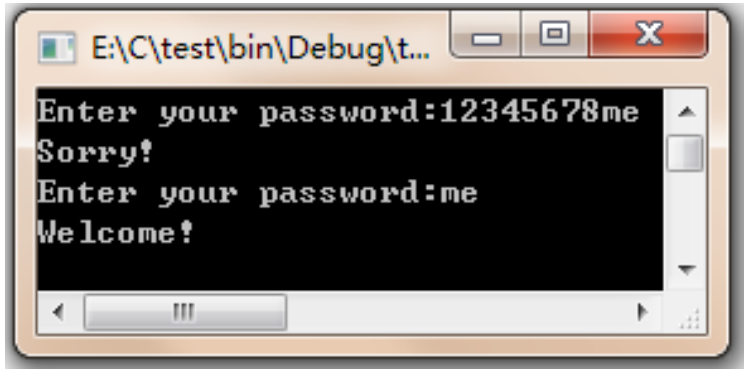
- 网络黑客常针对系统和程序自身存在的漏洞，编写相应的攻击程序
  - 对缓冲区溢出漏洞的攻击——最常见
  - 几乎占到了网络攻击次数的一半以上
- 世界上第一个缓冲区溢出攻击
  - Internet蠕虫——曾造成全球多台网络服务器瘫痪
- 何谓缓冲区溢出攻击？
  - 利用缓冲区溢出漏洞进行的攻击

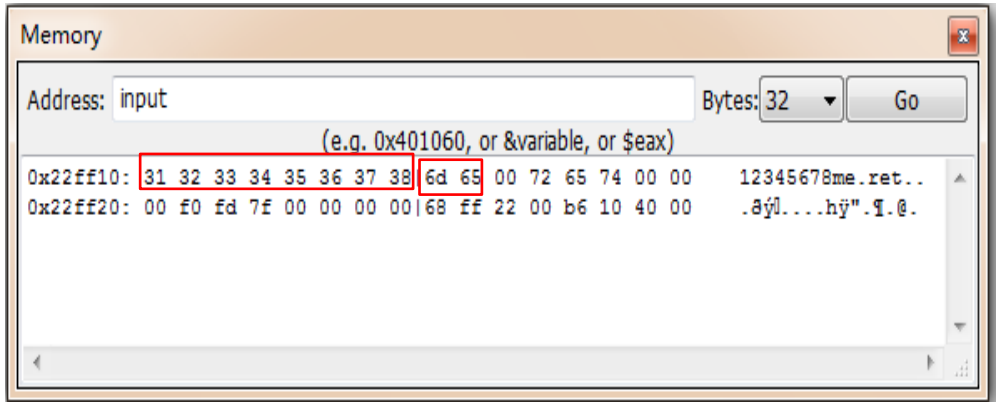
缓冲区溢出攻击

- 易引起缓冲区溢出攻击、不安全的函数
  - gets()、scanf()、strcpy()、strcat()等不限制字符串长度，不检查数组越界，易导致有用的堆栈数据被覆盖，给黑客攻击以可乘之机
- 对缓冲区溢出漏洞进行攻击的后果
  - 程序运行失败、系统崩溃和重启等
  - 利用缓冲区溢出，执行非授权指令，甚至取得系统特权，进而进行各种非法操作
- 防止和检测缓冲区溢出攻击
  - 成为防御网络入侵和入侵检测的重点之一

缓冲区溢出攻击实例

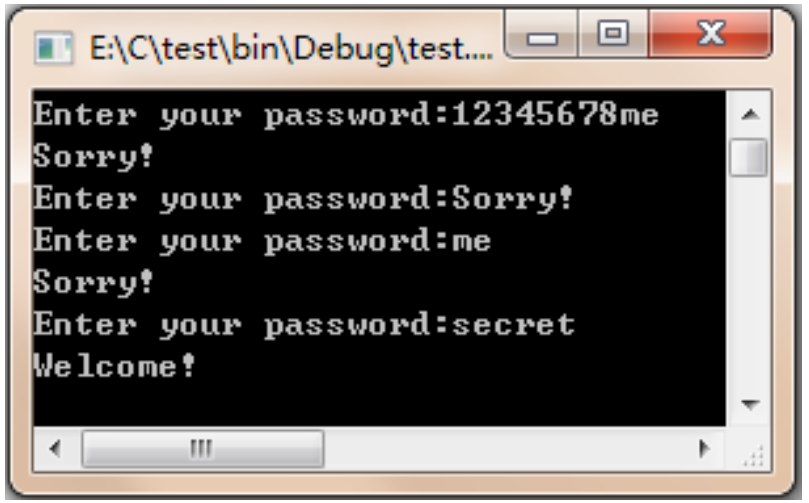
```
#include <stdio.h>
#include <string.h>
int main()
{
   char password[8] = "secret", input[8];
   while (1)
   {
     printf("Enter your password:");
     gets(input);
     if (strcmp(input, password) ==0)
     {
       printf("Welcome!\n");
       break;
     }
     else
    {
       printf("Sorry!\n");
     }
   }
   return 0;
}
```





字符串的安全输入方法

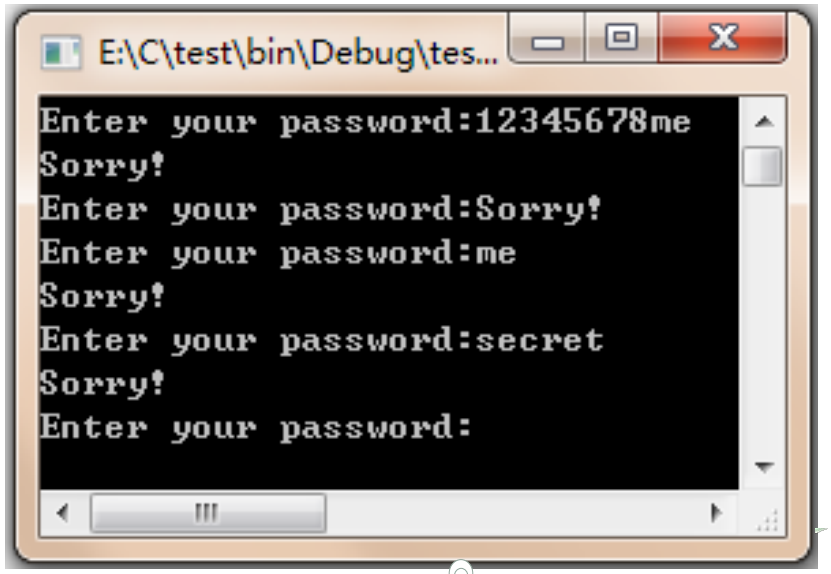
```
#include <stdio.h>
#include <string.h>
int main()
{
   char password[8] = "secret", input[8];
   while (1)
   {
     printf("Enter your password:");
     scanf("%7s", input);
     if (strcmp(input, password) ==0)
     {
       printf("Welcome!\n");
       break;
     }
     else
    {
       printf("Sorry!\n");
     }
   }
   return 0;
}
```



字符串的安全输入方法

```
#include <stdio.h>
#include <string.h>
int main()
{
   char password[8] = "secret", input[8];
   while (1)
   {
     printf("Enter your password:");
     fgets(input, sizeof(input), stdin);
     if (strcmp(input, password) ==0)
     {
       printf("Welcome!\n");
       break;
     }
     else
    {
       printf("Sorry!\n");
     }
   }
   return 0;
}
```

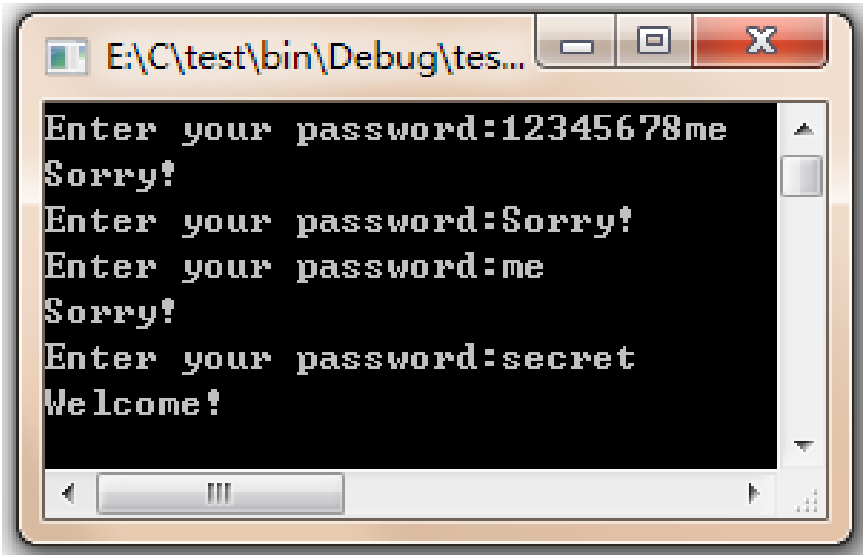
限制输入字符串的长度



字符串的安全输入方法

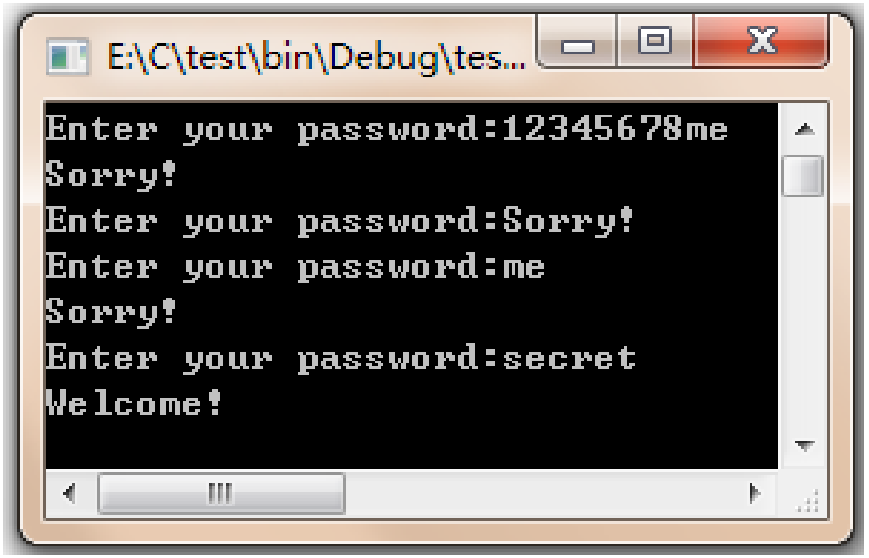
```
#include <stdio.h>
#include <string.h>
int main()
{
   char password[8] = "secret\n", input
   while (1)
   {
     printf("Enter your password:");
     fgets(input, sizeof(input), stdin);
     if (strcmp(input, password) ==0)
     {
       printf("Welcome!\n");
       break;
     }
     else
     {
       printf("Sorry!\n");
     }
   }
   return 0;
}
```

gets输入字符串时，读到'\n'时，
将'\n'换成空字符'\0'来存储
但fgets输入的是以'\n'为结尾的字符串



```c
#include <stdio.h>
#include <string.h>
int main()
{
   char password[8] = "secret", input[8];
   int i;
   while (1)
   {
     printf("Enter your password:");
     fgets(input, sizeof(input), stdin);
     for (i=0; input[i]!='\n'; i++)
     {
       ;
     }
     input[i] = '\0';
     if (strcmp(input, password) ==0)
     {
       printf("Welcome!\n");
       break;
     }
     else
    {
       printf("Sorry!\n");
     }
   }
   return 0;
}
```

将fgets输入的'\n'换成'\0'



防止缓冲区溢出的两个要点

- 使用更安全的字符串处理函数
  - 用fgets()、strncpy()、strncat()代替gets()、strcpy()、strcat()等不限制字符串长度，不检查数组越界的函数
- 在向一块内存中写入数据之前确认这块内存是否可以写入，同时检查写入的数据是否超过这块内存的大小