

第9章

网络攻击技术

主要内容

- ❖ 侦查
- ❖ 扫描
- ❖ 获取访问权限
- ❖ 保持访问权限
- ❖ 消除入侵痕迹
- ❖ 拒绝服务攻击

概述

- ❖ 网络攻击技术是一把双刃剑。对攻击者而言，它是攻击技术，对安全人员来说，它是不可缺少的安全防护技术。
- ❖ 对系统的攻击和入侵，都是利用软件或系统漏洞进行。
- ❖ 大多数情况下，恶意攻击者使用的工具和安全技术人员是相同的，这意味着安全人员必须了解攻击者使用的工具和手段

一次完整的网络攻击包含的基本步骤

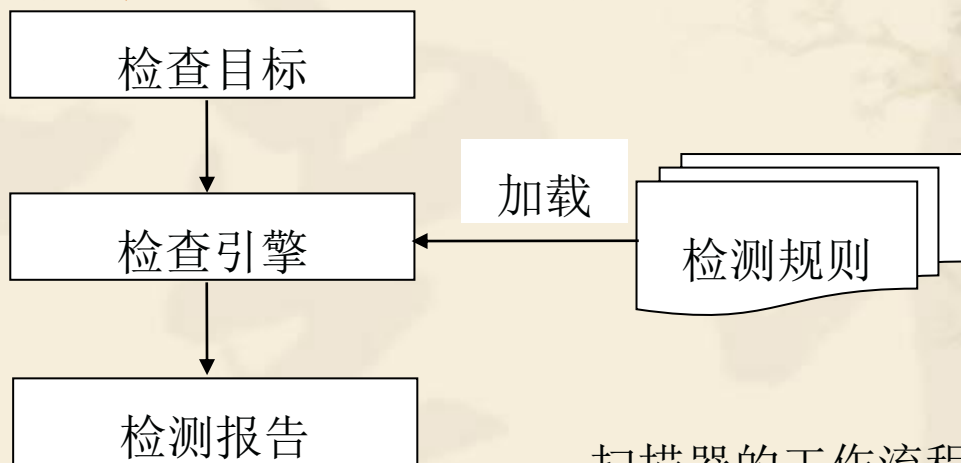
攻击的步骤	解释	例子
侦查	被动或者主动获取信息的过程	嗅探网络流量，察看HTML代码，社交工程，获取目标系统的一切信息
扫描及漏洞分析	识别所运行系统和系统上活动的服务，从中找出可攻击的弱点	Ping扫描、端口扫描、漏洞扫描
获取访问权限	攻击识别的漏洞，以获取未授权的访问权限	利用缓冲区溢出或者暴力破解口令，并登录系统
保持访问权限	上传恶意软件，以确保能重新进入系统	在系统上安装后门
消除痕迹	消除恶意活动的踪迹	删除或修改系统和应用日志中的数据

9.1 侦查

- ❖ 侦查也被称为踩点，目的是发现目标
- ❖ 通过踩点主要收集以下可用信息：
 - ❧ 网络域名
 - ❧ 内部网络
 - ❧ 外部网络
 - ❧ 目标所用的操作系统

9.2 扫描

- ❖ 扫描主要是指通过固定格式的询问来试探主机的某些特征的过程，而提供了扫描功能的软件工具就是扫描器。
- ❖ “一个好的扫描器相当于数百个合法用户的账户信息”。
- ❖ 分为端口扫描和漏洞扫描



扫描器的工作流程

端口扫描

- ❖ 扫描端口的主要目的是判断目标主机的操作系统以及开放了哪些服务。
- ❖ 端口是由计算机的通信协议TCP/IP协议定义的。计算机之间的通信，归根结底是进程间的通信，而端口则与进程相对应。
- ❖ 端口分类：
 - ❧ 熟知端口（公认端口）：由因特网指派名字和号码公司ICANN负责分配给一些常用的应用层程序固定使用的熟知端口，端口号一般为0~1023。表9-2和表9-3列出了常见的熟知端口。
 - ❧ 一般端口：用来随时分配给请求通信的客户进程。

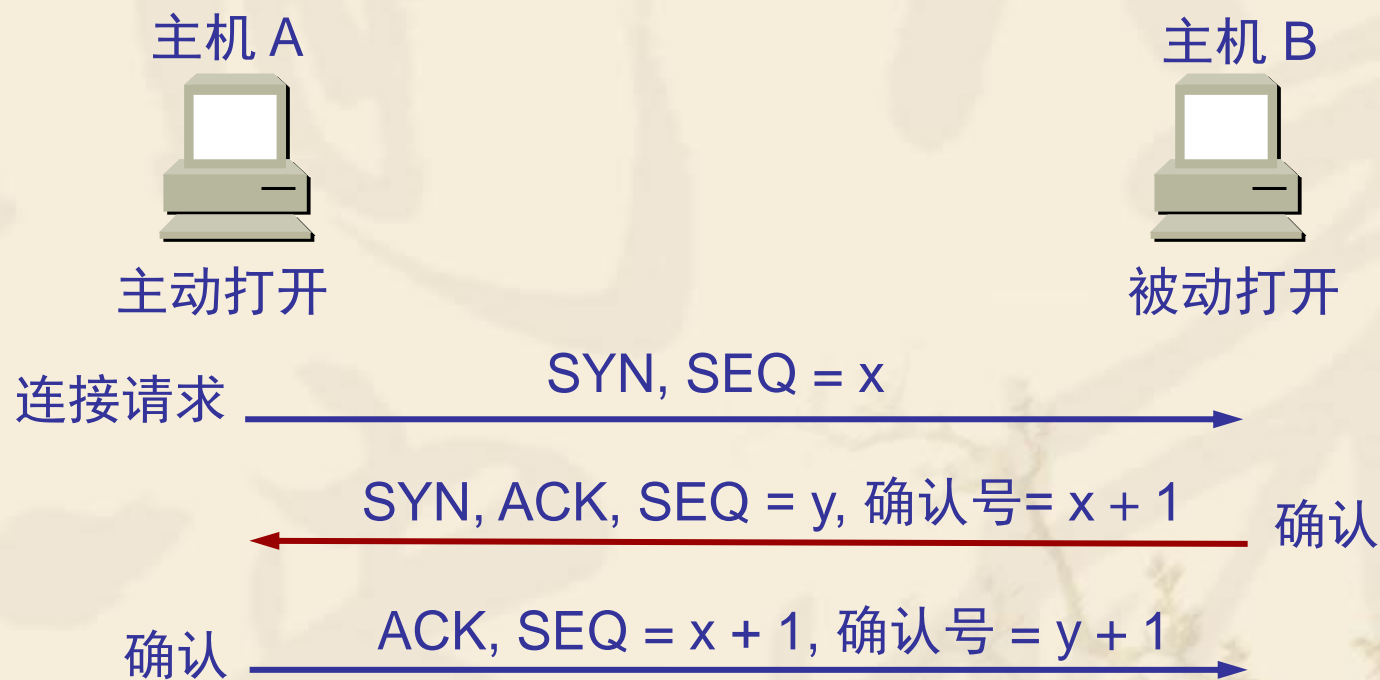
常见TCP熟知端口

服务名称	端口号	说明
FTP	21	文件传输服务
Telnet	23	远程登录服务
HTTP	80	网页浏览服务
POP3	110	邮件服务
SMTP	25	简单邮件传输服务
Socks	1080	代理服务

常见UDP熟知端口

服务名称	端口号	说明
RPC	111	远程调用
SNMP	161	简单网络管理
TFTP	69	简单文件传输

用三次握手建立 TCP 连接



扫描方法

❖ 全TCP连接

- ❧ 这种扫描方法使用三次握手，与目标计算机建立标准的TCP连接。
- ❧ 很容易被目标主机发觉并记录。

❖ 半打开式扫描

- ❧ 扫描主机自动向目标计算机的指定端口发送SYN数据段，表示发送建立连接请求，如果目标计算机的回应TCP报文中SYN=1，ACK=1，则说明该端口是活动的，接着扫描主机传送一个RST给目标主机拒绝建立TCP连接，从而导致三次握手过程的失败。
- ❧ 由于扫描过程中全连接尚未建立，所以大大降低了被目标计算机的记录的可能性

❖ FIN扫描

- ❧ 发送一个FIN=1的TCP报文到一个关闭的端口时，该报文会被丢掉，并返回一个RST报文。但是，如果当FIN报文发到一个活动的端口时，该报文只是简单的丢掉，不会返回任何回应。
- ❧ 扫描没有涉及任何TCP连接部分，因此，这种扫描比前两种都安全，可以称之为秘密扫描。

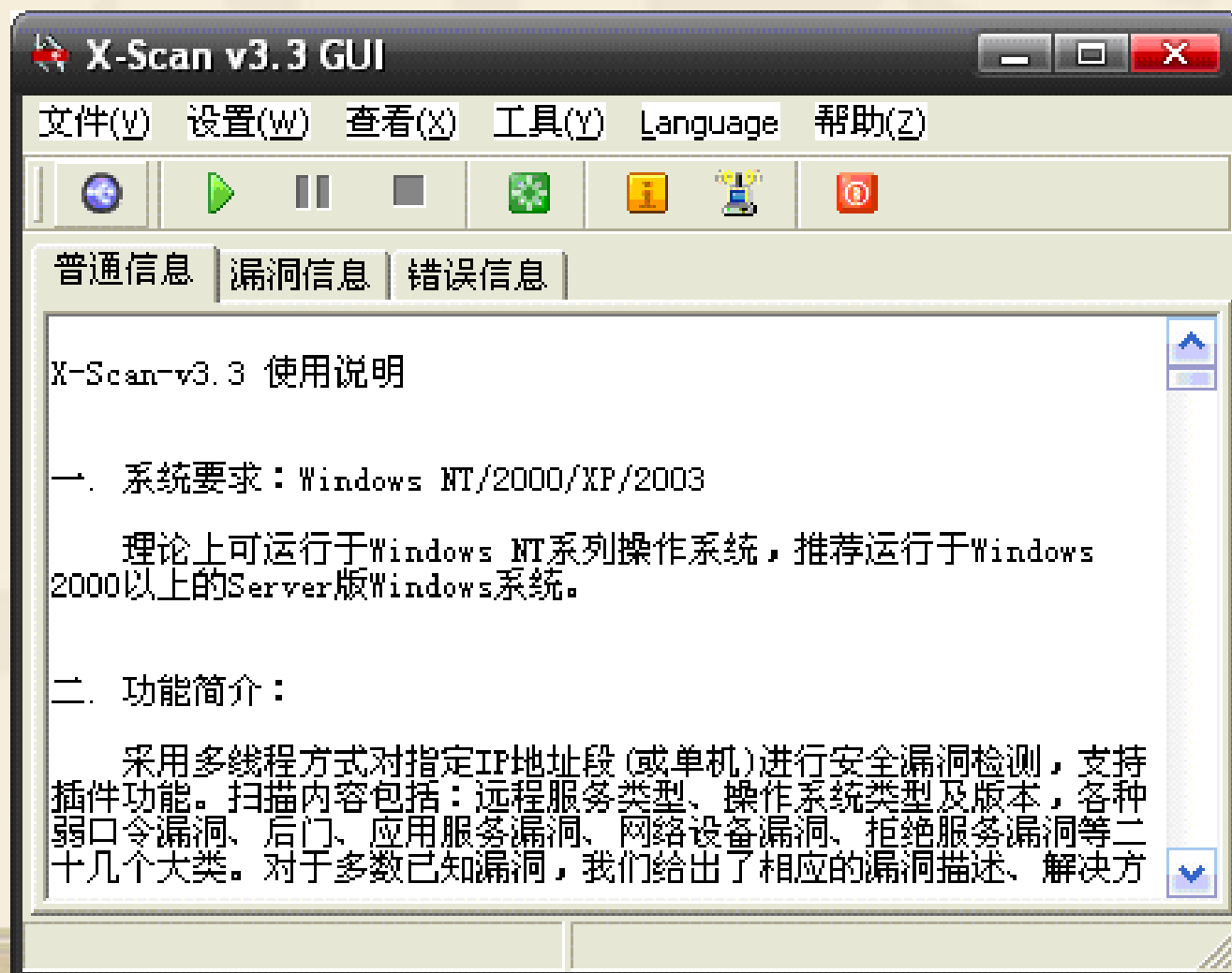
❖ 第三方扫描

- ❧ 代理扫描”，这种扫描是控制第三方主机来代替入侵者进行扫描
- ❧ 更加隐蔽

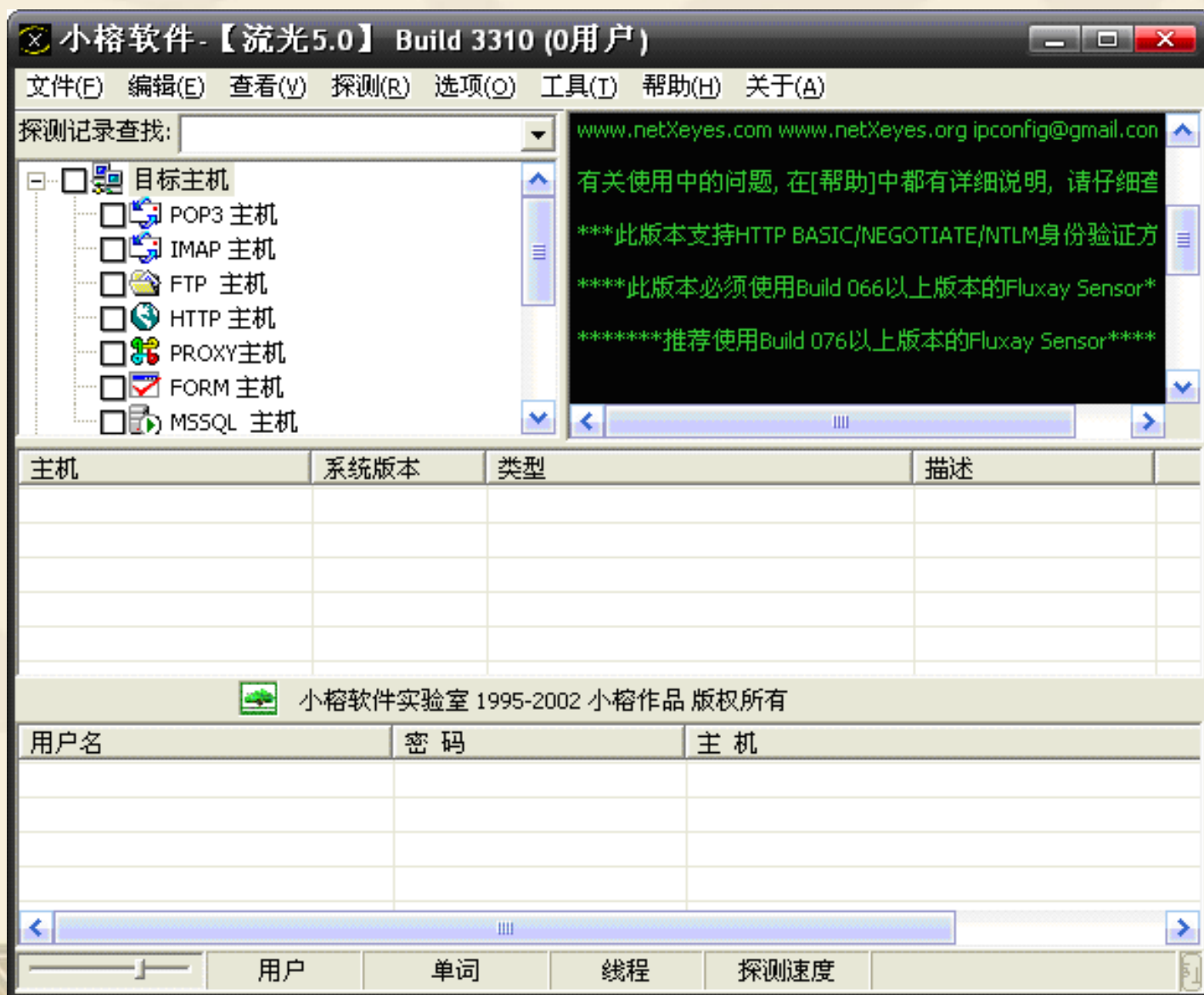
漏洞扫描

- ❖ 在各种各样的软件逻辑漏洞中，有一部分会引起非常严重的后果，我们把会引起软件做一些超出设计范围事情的bug称为漏洞。
- ❖ 为了防范这种漏洞，最好的办法是及时为操作系统和服务打补丁。所谓补丁，是软件公司为已发现的漏洞所作的修复行为。
- ❖ 漏洞扫描通常通过漏洞扫描器来执行。漏洞扫描器是通过在内部放置已知漏洞的特征，然后把被扫描系统特征和已知漏洞相比对，从而获取被扫描系统漏洞的过程。
- ❖ 漏洞扫描只能找出目标机上已经被发现并且公开的漏洞

扫描器X-Scan



流光



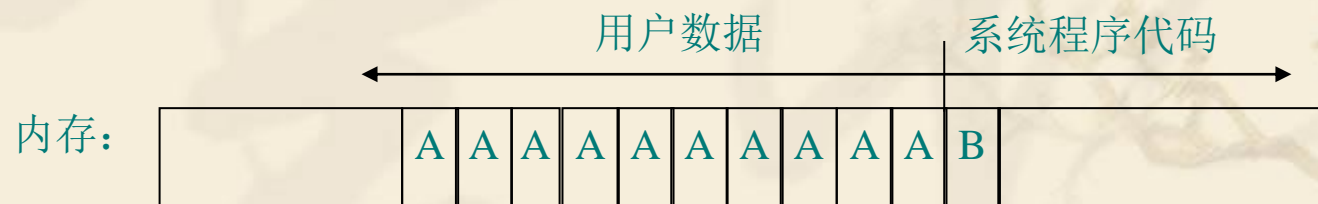
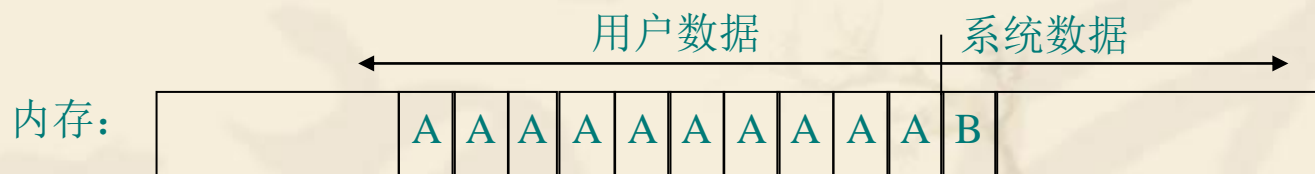
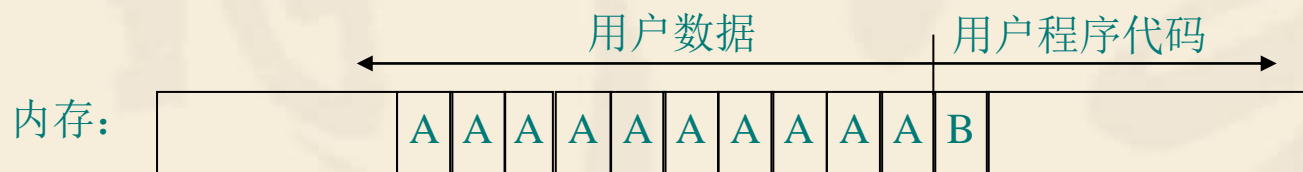
9.3 获取访问权限

- ❖ 缓冲区溢出
- ❖ SQL注入攻击

缓冲区溢出

- ❖ 缓冲区溢出就好比是把2升的水倒入1升的水罐中，肯定会有一部分水流出并且造成混乱。
- ❖ 声明一个在内存中占10个字节的字符串：
 ❧ `Char str1[10];`
- ❖ 编译器为缓冲区划分了10个字节大小的空间，从`str1[0]`到`str1[9]`，每个都分别占用一个字节的空間。
- ❖ 那么，执行下列语句：
 ❧ `Strcpy(str1, 'AAAAAAAAAAAAAAAAAAAAA');`
- ❖ 则会造成缓冲区溢出。因为`strcpy`这个函数根本就不会检查后面的字符串是否比`str1`所分配的空间要大。

如果溢出的数据为B，则可能：

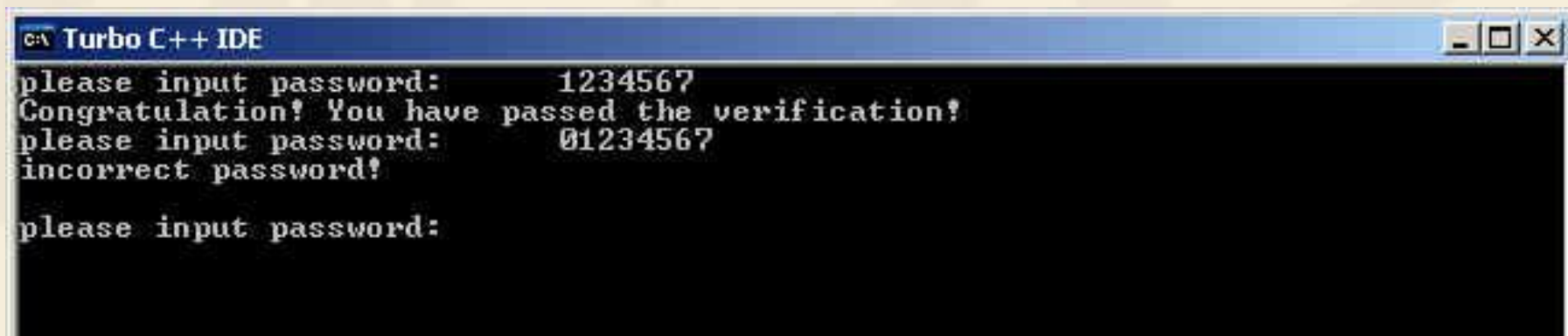


溢出实例

```
#include <stdio.h>
#define PASSWORD "1234567"
int verify_password (char *password)
{
    int authenticated;
    char buffer[8];
    authenticated=strcmp(password,PAS
        SWORD);
    strcpy(buffer,password);
    return authenticated;
}
```

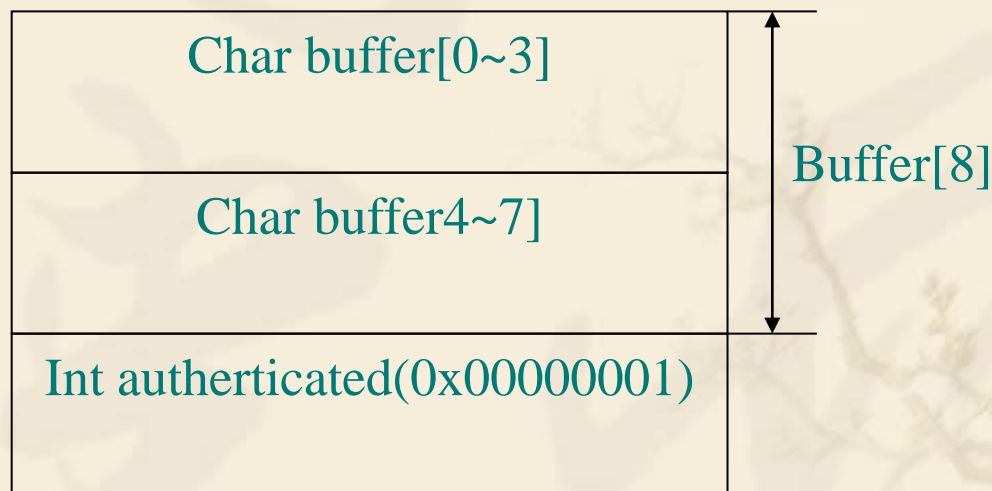
```
main()
{
    int valid_flag=0;
    char password[1024];
    while(1)
    {
        printf("please input password:   ");
        scanf("%s",password);
        valid_flag = verify_password(password);
        if(valid_flag)
        {
            printf("incorrect password!\n\n");
        }
        else
        {
            printf("Congratulation! You have passed
                the verification!\n");
            break;
        }
    }
}
```

输入了正确的密码“1234567”之后才能通过验证

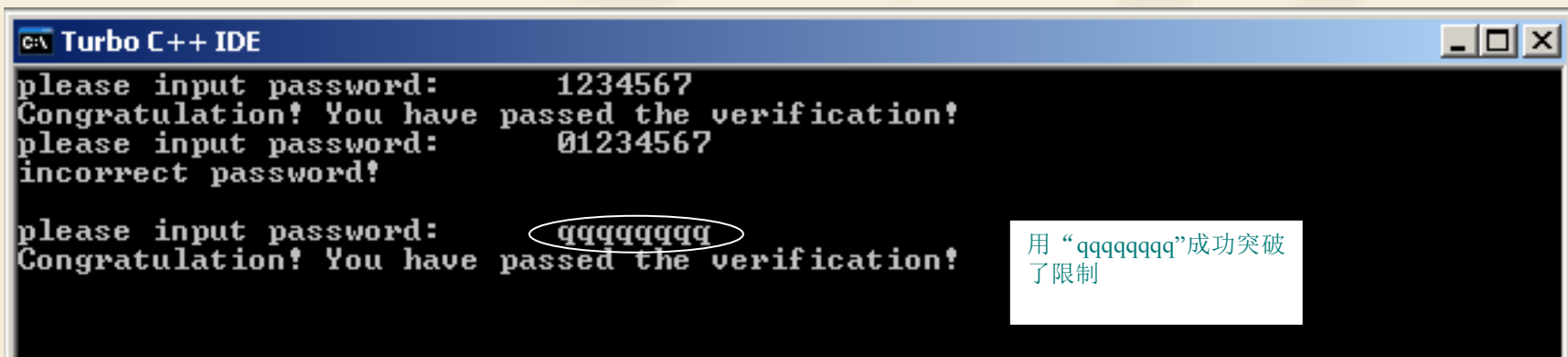
A screenshot of the Turbo C++ IDE window. The title bar reads "C:\ Turbo C++ IDE". The main window is black with white text. It shows a password verification process: first, the user enters "1234567" and receives a congratulatory message; second, the user enters "01234567" and receives an "incorrect password!" message; third, the prompt "please input password:" is shown again without any input or output yet.

```
C:\ Turbo C++ IDE  
please input password:      1234567  
Congratulation! You have passed the verification!  
please input password:      01234567  
incorrect password!  
please input password:
```


- ❖ 函数verify_password()里边申请了两个局部变量：int authenticated和char buffer[8]。当verify_password()被调用时，系统会给它分配一片连续的内存空间，这两个变量就分布在那里（实际上就叫函数栈帧）



- ❖ 输入包含8个字符的错误密码“qqqqqqqq”，那么buff[8]所拥有的8个字节将全部被“q”的ASCII码0x71填满，而字符串的结束标志NULL刚好写入了authenticated变量并且值为0x00000000。于是，错误的密码得到了正确密码的运行效果。



```
C:\ Turbo C++ IDE
please input password: 1234567
Congratulation! You have passed the verification!
please input password: 01234567
incorrect password!

please input password: qqqqqqqq
Congratulation! You have passed the verification!
```

用“qqqqqqqq”成功突破了限制

SQL注入攻击

- ❖ 如果用户的输入能够影响到脚本中**SQL**命令串的生成，那么很可能在添加了单引号、“#”号等转义命令字符后，能够改变数据库最终执行的**SQL**命令，攻击者就利用这个特点修改自己的输入，最终获取数据库中自己所需要的信息，例如管理员密码。这就是**SQL**注入。
- ❖ **SQL**注入是从正常的**WWW**端口访问，而且表面看起来跟一般的**Web**页面访问没什么区别，所以防火墙不会对**SQL**注入发出警报

9.4 保持访问权限

❖ 开放新帐号

- ❧ 容易被管理员察觉

❖ 安装后门

- ❧ 具有隐蔽性和非授权性

❖ Rootkit

- ❧ 通过替换系统文件来达到目的。这样就会更加的隐蔽，使检测变得比较困难。

9.5 消除入侵痕迹

- ❖ 在所有的操作系统中，包括Windows和Unix系统，都有自己的日志系统。在日志中记载了各种信息，例如用户对其的操作、应用程序所作的行为、各种各样的安全事件等。当入侵者非法进入了系统后，往往会被这些日志记录下来。而管理员通过日志，则有可能分析出入侵者的行为，甚至是根据线索找到入侵者。

Windows的“事件查看器”

- ❖ 通过“开始→控制面板→管理工具→事件查看器”将其打开



Windows的“事件查看器”

❖ 应用程序日志

☞ 由应用程序或系统程序记录的事件，主要记录程序运行方面的事件

❖ 安全性日志

☞ 记录了诸如有效和无效的登录尝试等事件，以及与资源使用相关的事件

❖ 系统日志

☞ Windows XP的系统组件记录的事件

Windows的日志文件

❖ DNS日志默认位置

❧ %systemroot%\system32\config, 默认文件大小512KB, 管理员都会改变这个默认大小;

❖ 安全日志文件

❧ %systemroot%\system32\config\SecEvent.EVT;

❖ 系统日志文件

❧ %systemroot%\system32\config\SysEvent.EVT;

❖ 应用程序日志文件

❧ %systemroot%\system32\config\AppEvent.EVT;

❖ FTP日志默认位置

❧ %systemroot%\system32\logfiles\msftpsvc1\, 默认每天一个日志;

❖ WWW日志默认位置

❧ %systemroot%\system32\logfiles\w3svc1\, 默认每天一个日志。

9.6 拒绝服务攻击

- ❖ 拒绝服务攻击（**Denial of Service**，简称**DoS**）是指攻击者利用系统的缺陷，通过执行一些恶意的操作而使合法的系统用户不能及时的得到服务或者系统资源，如**CPU**处理时间、存储器、网络带宽、**Web**服务等。
- ❖ 它本身并不能使攻击者获取什么资源，例如系统的控制权力、秘密的文件等，它只是以破坏服务为目的。

常见的拒绝服务攻击方法

常见的拒绝服务攻击方法

- ❖ 基于网络带宽消耗的拒绝服务攻击
- ❖ 消耗磁盘空间的拒绝服务攻击
- ❖ 消耗CPU资源和内存的拒绝服务攻击
- ❖ 基于系统缺陷的拒绝服务攻击

常见的拒绝服务攻击方法

❖ 基于网络带宽消耗的拒绝服务攻击

攻击者有意制造大量的数据包或传输大量文件以占用有限的带宽资源，使合法用户无法正常使用网络资源，从而实现攻击者的意图。

❖ 消耗磁盘空间的拒绝服务攻击

利用系统的缺陷，制造大量的垃圾信息。

❖ 消耗CPU资源和内存的拒绝服务攻击

利用系统的缺陷，有意使用大量的CPU和内存资源，从而导致系统服务性能下降甚至造成系统崩溃。

常见的拒绝服务攻击方法

❖ 基于系统缺陷的拒绝服务攻击

攻击者利用目标系统的漏洞和通信协议的弱点实现攻击。

分布式拒绝服务攻击

- ❖ 分布式拒绝服务（**Distributed Denial of Service, DDoS**）攻击指借助于客户/服务器技术，将多个计算机联合起来作为攻击平台，对一个或多个目标发动**DoS**攻击，从而成倍地提高拒绝服务攻击的威力。

