第3章 漏洞概念

学习要求：掌握漏洞的概念，了解按照漏洞生命周期的阶段对漏洞进行分类的方法；了解漏洞库的作用，掌握主要的漏洞库；掌握第一个漏洞示例。

课时：2课时。

3.1 概念及特点

3.1.1 概念

漏洞也称为脆弱性(Vulnerability)，是计算机系统的硬件、软件、协议在系统设计、具体实现、系统配置或安全策略上存在的缺陷。这些缺陷一旦被发现并被恶意利用，就会使攻击者在未授权的情况下访问或破坏系统，从而影响计算机系统的正常运行甚至造成安全损害。

漏洞概念视频

漏洞的定义包含了以下三个要素：首先，漏洞是计算机系统本身存在的缺陷；其次，漏洞的存在和利用都有一定的环境要求；最后，漏洞的存在本身是没有危害的，只有被攻击者恶意利用，才能给计算机系统带来威胁和损失。

长期以来，对于漏洞及其相关领域的概念有多种称呼，包括 Hole，Vulnerability，Error，Fault，Weakness，Failure 等，这些概念的含义不完全相同。Vulnerability 和 Hole 都是一个总的全局性概念，包括威胁、损坏计算机系统安全性的所有要素。Error 是指软件设计者或开发者犯下的错误，是导致不正确结果的行为，它可能是有意无意的误解、对问题考虑不全面所造成的过失等。Fault 则指计算机程序中不正确的步骤、方法或数据定义，是造成运行故障的根源。Weakness 指的是系统难以克服的缺陷或不足，缺陷和错误可以更正、解决，但不足和弱点可能没有解决的办法。Failure 是指执行代码后所导致的不正确的结果，造成系统或系统部件不能完成其必需的功能，也可以称为故障。

广义来讲，如果这些术语的使用不会引起误会，可以将错误、缺陷、弱点等上述称呼包含的内容都归为漏洞。不过由于漏洞是一个全面综合的概念，所以错误、缺陷、弱点和故障等并不等同于漏洞，而只是漏洞的一个方面。

如果从狭义的角度理解，漏洞是计算机系统中可以被攻击者利用从而对系统造成较大危害的安全缺陷。狭义的“漏洞”是广义的“漏洞”中可被利用并造成危害的那部分漏洞。

软件漏洞专指计算机系统中的软件系统漏洞。软件漏洞会涉及操作系统、数据库、应

用软件、应用服务器、信息系统、定制应用系统的安全,并且很多硬件系统中运行的软件也同样会造成不良后果,包括嵌入式设备、工业控制系统等,因而影响广泛。

3.1.2 特点

软件漏洞对软件的安全运行影响很大,主要表现在以下几个方面。

1. 软件漏洞危害性大

软件漏洞一旦被攻击者利用,会威胁软件系统的安全。软件漏洞的恶意利用能够影响网民的工作、生活,甚至为社会和国家带来灾难性后果。

2. 软件漏洞影响广泛

计算机系统中的软硬件都离不开软件程序,大多数硬件的正常运行也离不开硬件控制程序。因而,软件漏洞会影响绝大多数的软硬件设备,包括操作系统本身及其支撑软件,网络和服务器软件,网络路由器和安全防火墙等。

3. 软件漏洞存在的长久性

软件漏洞随着软件系统的发布而不断暴露出来,一个软件系统的漏洞会伴随这个系统整个生命周期。在推出新版系统纠正旧版本中漏洞的同时,也会引入一些新的漏洞和错误。因而随着时间的推移,漏洞问题也会长期存在。

4. 软件漏洞的隐蔽性

软件漏洞本身的存在没有危害,在通常情况下并不会对系统安全造成危害,只有被攻击者在一定条件下利用才会影响系统安全,因此这些软件漏洞具有很大的隐蔽性。

3.2 漏洞分类

漏洞的分类方法有很多种,下面从多个角度来区分不同类别的漏洞。此外,根据软件漏洞的危险级别,介绍软件漏洞的危险级别分级方法。

3.2.1 漏洞分类

软件漏洞可以考虑从以下几方面进行漏洞的分类,包括:软件漏洞被攻击者利用的地点,软件漏洞形成原因,漏洞生命周期不同阶段,漏洞对系统安全的直接威胁后果等。

1. 按照软件漏洞被攻击者利用的地点进行分类

(1) 本地利用漏洞。

本地利用漏洞是指攻击者必须在本机拥有访问权限的前提下才能攻击并利用的软件漏洞。比较典型的是没有网络服务功能的本地软件漏洞,以及本地权限提升漏洞,也叫本地提权漏洞。本地提权漏洞能让普通用户获得最高管理员权限甚至系统内核的权限。

(2) 远程利用漏洞。

远程利用漏洞是指攻击者可以直接通过网络发起攻击并利用的软件漏洞。这类软件

漏洞危害极大，攻击者能随心所欲地通过此漏洞对远端计算机进行远程控制，此类漏洞也是蠕虫病毒主要利用的漏洞。

2. 根据漏洞形成原因分类

根据漏洞触发原因的不同，可以将软件漏洞进行以下几种类别的划分。

(1) 输入验证错误漏洞。

输入验证错误漏洞是由于未对用户输入数据的合法性进行验证，使攻击者能利用漏洞非法进入系统。

(2) 缓冲区溢出漏洞。

缓冲区溢出漏洞是由于向程序的缓冲区中输入的数据超过其规定长度，造成缓冲区溢出，破坏程序正常的堆栈，使程序执行其他指令。

(3) 设计错误漏洞。

设计错误漏洞是由于程序设计错误而导致的漏洞。大多数的漏洞都可归类于设计错误。

(4) 意外情况处置错误漏洞。

意外情况处置错误漏洞是由于程序在实现逻辑中没有考虑到一些意外情况，而导致运行出错。

(5) 访问验证错误漏洞。

访问验证错误漏洞是由于程序的访问验证部分存在某些逻辑错误，使攻击者可以绕过访问控制进入系统。

(6) 配置错误漏洞。

配置错误漏洞是由于系统和应用的配置有误，或配置参数、访问权限、策略安装位置有误造成的。

(7) 竞争条件漏洞。

竞争条件漏洞是由于程序处理文件等实体时，在时序和同步方面存在缺陷，导致攻击者可以利用存在的机会窗口施以外来影响。

(8) 环境错误漏洞。

环境错误漏洞是由于一些环境变量的错误或恶意设置造成的漏洞。

(9) 外部数据被异常执行漏洞。

外部数据被异常执行漏洞是指攻击者在外部非法输入的数据，被系统作为代码解释执行，典型的有 SQL 注入和 XSS 等。

3. 根据漏洞生命周期不同阶段的分类

一个漏洞从被攻击者发现并利用，到被厂商截获并发布补丁，再到补丁被大多数用户安装导致漏洞失去了利用价值，一般都要经历一个完整的生命周期。按照漏洞生命周期的阶段进行分类的方法包括以下三种。

(1) 0day 漏洞。

0day 漏洞指还处于未公开状态的漏洞。

这类漏洞只在攻击者个人或者小范围黑客团体内使用，网络用户和厂商都不知情，因

此没有任何防范手段,危害非常大。

越来越多的破解者和黑客,已经把目光从率先发布漏洞信息的荣誉感转变到利用这些漏洞而得到的经济利益上,互联网到处充斥着数以万计的充满入侵激情的脚本小子,更不涉及那些以窃取信息为职业的商业间谍和情报人员了。于是,0day 漏洞有了市场。

国外多年前就有了 0day 漏洞的网上交易,黑客们通过网上报价出售手中未公开的漏洞信息,一个操作系统或数据库的远程溢出源码可以卖到上千美元甚至更高。而国内的黑客前不久也在网上建立了一个专门出售入侵程序号称中国第一 0Day 的网站,尽管类似提供黑客工具的网站很多,但此网站与其他网站的区别在于 0Day 的特性十分明显,价格较高的攻击程序的攻击对象,还没有相应的安全补丁,也就是说这样的攻击程序很可能具有一击必中的效果。这个网站成立不久便在搜索引擎中消失了,也许已经关闭,也许转入地下。但不管怎样,0Day 带来的潜在经济利益不可抹杀,而其将来对信息安全的影响以及危害也绝不能轻视。

(2) 1day 漏洞。

1day 漏洞原义是指补丁发布在 1 天内的漏洞,不过通常指发布补丁时间不长的漏洞。

由于了解此漏洞并且安装补丁的人还不多,这种漏洞仍然存在一定的危害。利用 1day 漏洞进行扩散的蠕虫及漏洞利用程序,趁着大量用户还未打补丁这个时间差,会攻击大批的计算机系统。

(3) 已公开漏洞。

已公开漏洞是指厂商已经发布补丁或修补方法,大多数用户都已打过补丁的漏洞。这类漏洞从技术上因为已经有防范手段,并且大部分用户已经进行了修补,危害比较小。

软件漏洞对计算机系统安全的威胁不仅限于单个软件漏洞的直接威胁,攻击者往往组合利用多个软件漏洞,最终得到远程目标计算机的最高权限。例如,获得系统普通用户访问权限后,攻击者有可能继续利用本地权限提升漏洞,将自己提升到系统管理员的权限。

3.2.2 危险等级划分

软件漏洞的危险等级划分,是根据软件漏洞在破坏性、危害性、严重性方面造成的潜在威胁程度,以及漏洞被利用的可能性,对各种软件漏洞进行分级。软件漏洞危险等级划分的方法很多,一般被分为高危、中危、低危三个危险级别,或者进一步细分为四个危险级别。下面介绍一下紧急、重要、中危、低危四个级别的划分。

1. 第一级:紧急

这是危险级别最高的等级,紧急级别漏洞的利用可以导致网络蠕虫和病毒在用户不知情的情况下在网络上任意传播和繁殖,或者导致执行远程恶意代码。这类漏洞可被攻击者进行利用的软件环境较为普遍,并且适用的操作系统较广泛。对于紧急级别的漏洞,需要立即安装升级包(补丁程序)。微软公司安全公告的第一级定义为“严重”。

2. 第二级:重要

对重要级别漏洞的利用可以导致严重的后果,包括导致执行远程恶意代码,或者导致

权限提升。这类漏洞可被攻击者进行利用的软件环境和适用的操作系统比第一级要苛刻，利用的限制较多，但也可以危害到用户数据和相关资源的机密性、完整性和有效性。对于被评为重要级别的安全漏洞，需要安装升级包（补丁程序）。微软公司安全公告的第二级定义为“重要”。

3. 第三级：中危

对于中危级别漏洞，由于默认配置、审核或难以利用因素的影响，中危级别漏洞的利用效果显著降低，可以危害到用户数据和相关资源的可用性、完整性和有效性。这类漏洞包括拒绝服务攻击漏洞等。针对中危级别的漏洞，可以安装升级包。微软公司安全公告的第三级定义为“中等”。

4. 第四级：低危

低危级别漏洞的利用难度非常大，其利用的效果已经起不到危害用户数据的可用性、完整性，或者已降至最低限度。关于低危系统安全漏洞则应该在阅读安全信息以后判断该安全漏洞是否对此系统产生影响。针对低危级别的漏洞，可以安装升级包。微软公司安全公告的第四级定义为“警告”。

3.3 漏洞库

随着计算机软件技术的快速发展，大量的软件漏洞需要一个统一的命名和管理规范，以便开展针对软件漏洞的研究，提升漏洞的检测水平，并为软件使用者和厂商提供有关软件漏洞的确切信息。在这种需求推动下，多个机构和相关国家建立了漏洞数据库，这些数据库分为公开的和某些组织机构私有的不公开数据库。公开的数据库包括 CVE、NVD、BugTraq、CNNVD、CNVD 等。除了这些软件漏洞的公开来源外，还应该存在着大量的没有对公众开放的漏洞数据库。例如，IBM 建立的内部专用漏洞库 Vulda 等。

通过这些漏洞信息数据库，可以从中找到操作系统和应用程序的特定版本所包含的漏洞信息，有的还提供针对某些漏洞的专家建议、修复办法和专门的补丁程序，极少的漏洞库还提供检测、测试漏洞的 POC（Proof-Of-Concepts，为观点提供证据）样本验证代码。

目前，为了应对软件漏洞的威胁，许多国家建立了针对漏洞的应急响应机构，例如美国计算机应急反应小组（United States Computer Emergency Readiness Team，US-CERT）。应急小组已发展到许多国家，包括德国、澳大利亚等国家，以及中国的国家互联网应急中心 CNCERT/CC。他们是软件漏洞数据的主要提供者或者漏洞库的主要维护者，并且提供了高风险的漏洞警报和专家建议。另外，还有许多不同的国家官方组织、规模较大的企业和专门从事 IT 安全领域研究的机构和企业，在很大程度上推动了漏洞数据库领域的研究工作。

下面介绍一些国内外的著名漏洞数据库。

3.3.1 CVE

MITRE是一个受美国资助的基于麻省理工学院科研机构形成的非营利公司。MITRE公司建立的通用漏洞列表CVE(Common Vulnerabilities and Exposures)相当于软件漏洞的一个行业标准。它实现了安全漏洞命名机制的规范化和标准化,为每个漏洞确定了唯一的名称和标准化的描述,为不同漏洞库之间的信息录入及数据交换提供了统一的标识,使不同的漏洞库和安全工具更容易共享数据,成为评价相应入侵检测和漏洞扫描等工具和数据库的基准。

CVE中软件漏洞条目的命名过程,首先是CVE编委从一些讨论组、软件商发布的技术文件和一些个人或公司提供的资料中找到存在的安全问题,然后会给这种安全问题分配一个CVE候补名称,即CAN名称,相关的信息也会按照CVE条目的格式写成一个CAN条目(CVE Candidate Entry)。如果经过CVE编委讨论并投票通过,CAN条目就成为正式的CVE条目,在条目名称上只是相应地把CAN改成CVE。

现在有大量的公司和组织宣布他们的产品或数据库是CVE兼容的,如Security Focus Vulnerability Database、CERT/CC Vulnerability Notes Dalebaselb、X-Force Database、Cisco Secure Intrusion Detection System、BUGTPAQ。所谓与CVE兼容就是能够利用CVE中漏洞名称同其他CVE兼容的产品进行交叉引用。也就是通过CVE为每个漏洞分配的唯一名称,在其他使用了这个名称的工具、网站、数据库和服务中检索到相关信息,同时自身关于该漏洞的信息也能够被它们所检索。

CVE漏洞库的链接为http://www.cve.mitre.org。

3.3.2 NVD

美国国家漏洞数据库NVD(National Vulnerabilities Database)是美国国家标准与技术局NIST于2005年创建的,由国土安全部DHS的国家赛博防卫部和US-cert赞助支持。

NVD同时收录三个漏洞数据库的信息:CVE漏洞公告、US-CERT漏洞公告和US-CERT安全警告。同时,自己发布的漏洞公告和安全警告,是目前世界上数据量最大,条目最多的漏洞数据库之一。NVD漏洞库与CVE是同步和兼容的,CVE发布的新漏洞都会同步到NVD漏洞库中。所以,NVD能够第一时间发布最新的漏洞公告,信息发布的速度非常快。NVD数据库条目非常多且信息准确可靠,所以信息权威性非常高。

NVD的网址是http://nvd.nist.gov/。

3.3.3 CNNVD

中国国家信息安全漏洞库CNNVD(China National Vulnerability Database of Information Security)隶属于中国信息安全测评中心,是中国信息安全测评中心为切实履行漏洞分析和风险评估的职能,负责建设运维的国家级信息安全漏洞库,为我国信息安全保障提供基础服务。

CNNVD 信息安全漏洞定向通报服务是测评中心面向各级政府机关及企事业单位，及时、准确推送涵盖以漏洞信息为核心的各类数据及应用服务，主要包括定期向委托方提供与委托方相关的高危信息安全漏洞的分析及整改方案等，通过定期的信息安全漏洞信息通报、态势分析报告、研究报告及技术培训与咨询等途径，帮助委托方及时发现并排除自身的信息安全隐患，降低信息安全事件发生的可能性，提高委托方信息安全威胁应对与风险管理的能力和水平。

CNNVD 的网址为 http://www.cnnvd.org.cn。

3.3.4　CNVD

国家互联网应急中心(CNCERT 或 CNCERT/CC)成立于 1999 年 9 月，是工业和信息化部领导下的国家级网络安全应急机构。国家信息安全漏洞共享平台 CNVD(China National Vulnerability Database)是 CNCERT 联合国内重要信息系统单位、基础电信运营商、网络安全厂商、软件厂商和互联网企业建立的信息安全漏洞信息共享知识库，致力于建立国家统一的信息安全漏洞收集、发布、验证、分析等应急处理体系。

CNCERT 通过 CNVD 进行漏洞的收集整理、验证和漏洞库的建设，处理国内重要软件厂商、互联网厂商的漏洞安全事件，面向基础信息网络、重要信息系统和社会公众提供包括漏洞和补丁公告、漏洞趋势统计分析，并提供相应的应急响应和技术支撑服务。

CNVD 网站为 http://www.cnvd.org.cn。

3.3.5　BugTraq

安全焦点(SecurityFocus)是赛门铁克公司 2002 年收购的一个著名安全网站，被认为是互联网上最广泛且最可信的安全信息源，其所包含的漏洞列表是目前最大的漏洞数据库之一。SecurityFocus 的 BugTraq 邮件列表是整个安全社区重要的信息来源和技术讨论区，很多最新的技术讨论都发布在这里，很多 0day 的漏洞也往往首先出现在这里。它几乎包含所有的漏洞信息，并与 CVE 交叉索引，其漏洞编号 BID 是其他漏洞数据库必须引用的参考资料之一。由于访问量众多，其更新速度非常快。SecurityFocus 发布的漏洞信息中，提供了部分漏洞的利用程序下载，这在其他漏洞数据库中很少见。SecurityFocus 的漏洞库具有高级检索功能，可以按照厂商、标题、产品版本进行漏洞检索。同时，还可以根据 CVE 编号检索 CVE 的漏洞信息。

BugTraq 的网址为 http://www.securityfocus.com/。

3.3.6　其他漏洞库

除了上述权威漏洞数据库以外，多个安全组织机构和企业也发布自己建立的漏洞库和漏洞公告信息。

1. EDB 漏洞库

EDB(Exploit Database)漏洞库是由十多位安全技术人员志愿维护的数据库，包含

了大量软件的漏洞攻击代码。不同于只提供安全公告和建议的安全网站,这个漏洞库是一个包含了大量免费使用的攻击代码和POC样本验证代码的开放资源库。这些攻击代码和POC是通过直接提交、邮件列表和其他开放资源收集的。它为渗透测试人员、漏洞研究人员进行漏洞挖掘和利用研究提供了极大的帮助,而且EDB和CVE是兼容的。此外,这个漏洞库网站中包含了谷歌黑客数据库GHDB(Google Hacking Database),这个数据库中包含了很多搜索词,可以利用谷歌的搜索引擎直接搜索这些包含安全缺陷脚本的搜索词,就可以直接搜索到有缺陷的网站,从而可以让渗透者更快地了解一个网站应用程序中是否存在可利用的攻击代码。

EDB漏洞库的网址是http://www.exploit-db.com/。

2. 微软安全公告板和微软安全建议

微软安全公告板和微软安全建议中包括了与微软关联的安全漏洞信息,是用户获取Windows系列操作系统和微软应用程序相关漏洞的最权威、最详细的信息来源。

网址为http://www.microsoft.com/china/technet/security/current.mspx。

3. 绿盟科技的中文安全漏洞库

绿盟科技的中文安全漏洞库是目前国内漏洞数量最多、更新最快的漏洞数据库之一。截至2013年2月,漏洞数据库条目已超过2.2万条。

网址为http://www.nsfocus.net/vulndb。

4. 启明星辰的中文安全公告库

启明星辰是国内较早开始进行漏洞研究的厂商之一,它的中文安全公告库以安全公告的形式提供漏洞信息。

网址为http://www.venustech.com.cn/。

3.4 第一个漏洞

3.4.1 漏洞示例

下面的VC6程序演示了一个溢出漏洞,代码如示例3-1所示。

示例 3-1

```
#include <stdio.h>;
#include <stdlib.h>;
//Have we invoked this function?
void why_here(void)
{
    printf("why u r here?!\n");
    exit(0);
```

```
}
void f()
{
    int buff[1];
    buff[2] = (int)why_here;
}
int main(int argc, char * argv[])
{
    f();
    return 0;
}
```

如程序所示，主函数将调用函数 f，并没有调用 why_here 函数，运行结果如图 3-1 所示。

图 3-1　运行结果

> **注意**：如果是 Vs2005 及以上版本，需要代码做一下修改：buff[3] =(int)why_here;
>
> 在学习漏洞原理的例子中，我们通常采用 VC6 来编译代码，原因是 VC6 中没有增加针对溢出等漏洞的防护机制，没有更为直观地学习和观察漏洞出现的过程。

为什么运行结果是这样呢？

出现此类运行结果的根本原因是发生了缓冲区溢出。在函数 f 中，所声明的数组 buff 长度为 1，但是由于没有对访问下标的值进行校验，程序中对数组外的内存进行了读写。观察函数 f 的局部变量 buff 的内存示意。buff 是静态数组，buff 的值就是数组在内存的首地址。而 inf buff[1]意味着开辟了一个 4 字节的整数数组的空间，如图 3-2 所示。

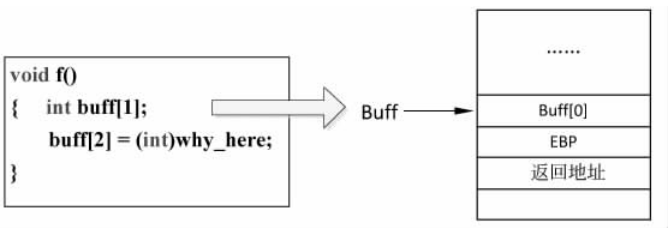

图 3-2　整数数组的空间

函数的栈区中,局部变量区域存的是数组元素 buff[0]的值。而 buff[2]则指向了返回地址。而 buff[2]赋值为 why_here,意味着返回地址被写入了 4 字节的函数 why_here 的地址。这样,在函数 f 执行完毕恢复到主函数 main 继续运行时,因为返回地址被改写成 why_here 函数的地址,而覆盖了原来的主函数 main 的下一条指令的地址,因此,发生了执行跳转。这是一个典型的溢出漏洞。

思考一个问题,如果将程序进行修改,请填空。

```
#include <stdio.h>;
#include <stdlib.h>;
//Have we invoked this function?
void why_here(void)
{
    printf("why u r here?!\n");
    exit(0);
}
void f()
{
    int buff ;
    int * p = &buff;
    _____ = (int)why_here;
}
int main(int argc, char * argv[])
{
    f();
    return 0;
}
```

答案:*(p+2)或者 p[2],原因是什么呢? p 的地址是变量 buff 的地址,而返回地址是&buff+8 的位置,采用指针表示,其地址就是 p+2,所以是*(p+2)或者 p[2]。

实验 1:使用 OllyDBG 运行该程序,观察栈帧变化情况,特别是返回地址的数值在调用函数 f 之前和调用 f 时的变化,以进一步掌握 OllyDBG 的使用。

3.4.2 漏洞利用示例

在第 2 章中,通过修改代码实现了软件破解,接下来将通过输入方面,试着覆盖临近变量的值,以便更改程序执行流程。

函数的局部变量在栈中一个挨着一个排列,如果这些局部变量中有数组之类的缓冲区,并且程序中存在数组越界的缺陷,那么越界的数组元素就有可能破坏栈中相邻变量的值,甚至破坏栈帧中所保存的 EBP 值、返回地址等重要数据。

用一个非常简单的例子(示例 3-2)来说明破坏栈内局部变量对程序的安全性有什么影响。

示例 3-2

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

```
#define PASSWORD "1234567"
int verify_password(char * password)
{
    int authenticated;
    char buffer[8]; //add local buff to be overflowed
    authenticated = strcmp(password, PASSWORD);
    strcpy(buffer, password);
    return authenticated;
}
void 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;
        }
    }
}
```

在 verify_password 函数的栈帧中，局部变量 int authenticated 恰好位于缓冲区 char buffer[8]的“下方”。

authenticated 为 int 类型，在内存中是一个 DWORD，占 4 字节。所以，如果能够让 buffer 数组越界，buffer[8]、buffer[9]、buffer[10]、buffer[11]将写入相邻的变量 authenticated 中。

观察一下源代码不难发现，authenticated 变量的值来源于 strcmp 函数的返回值，之后会返回给 main 函数作为密码验证成功与否的标志变量：当 authenticated 为 0 时，表示验证成功；反之，验证不成功。

如果用户输入的密码超过了 7 个字符(注意，字符串截断符 NULL 将占用 1 字节)，则越界字符的 ASCII 码会修改掉 authenticated 的值。如果这段溢出数据恰好把 authenticated 改为 0，则程序流程将被改变。

> **注意**：上述的程序基于 VC6 来编译执行才可以成功。

要成功覆盖临近变量并使其为 0,有两个条件:第一,输入一个 8 位的字符串的时候,如 22334455,此时字符串的结束符恰恰是 0,则覆盖变量 authenticated 的高字节并使其为 0;第二,输入的字符串应该大于 12345678,因为执行 strcmp 之后要确保变量 authenticated 的值为 1,也就是说只有高字节是 1,其他字节为 0。

实验 2:研究怎样用非法的超长密码去修改 buffer 的邻接变量 authenticated 从而绕过密码验证程序这样一件有趣的事情。

如果使用 Vs2005 或以上版本来调试该程序,发现上述过程无法成功,原因是什么呢?请使用 OllyDBG 观察栈区内存结构变化,确认是否在 buffer 和变量中间增加了一些随机数?