

第一章 安全基础技术实验

1.1 实验要求

利用西普信息安全教育平台，从以下课程部分选取 2 个实验项目进行学习、理解、设计和完成实验，书写所完成的内容。

1.2 实验内容

- 密码学及应用
- 防火墙 (*)
- 入侵检测系统
- 日志数据分析
- 数据库安全 (*)

1.3 防火墙

1.3.1 实验目的-搭建 iptables 实验环境

搭建 iptables 实验环境.

1.3.2 实验原理

VMware Workstation 软件包含一个用于英特尔 x86 兼容计算机的虚拟机套装，其允许多个 x86 虚拟机同时被创建和运行。每个虚拟机实例可以运行其自己的客户机操作系统。VMware 工作站允许一台真实的计算机同时运行数个操作系统。

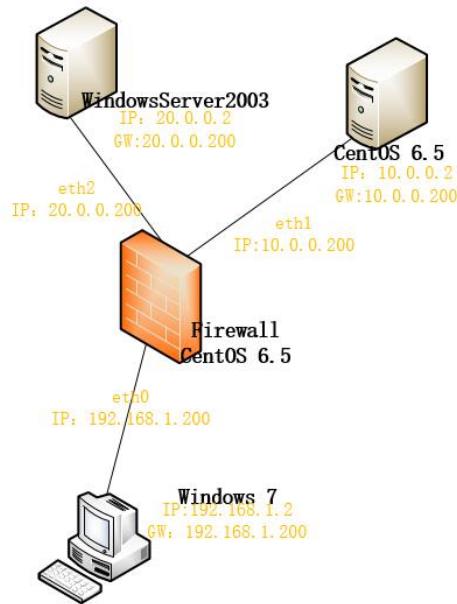
1.3.3 实验环境

软件：

- VMware
- 操作系统：

- Windows 7
- Windows server 2003
- CentOS 6.5
- CentOS 6.5

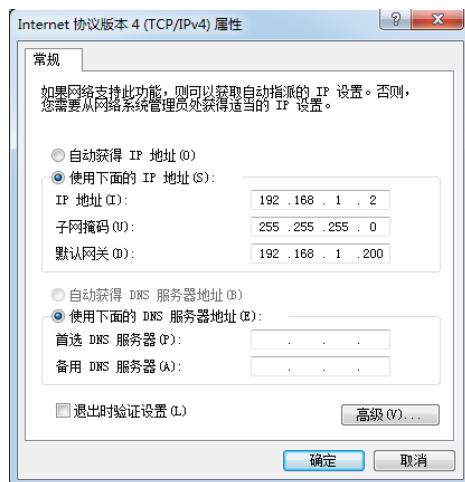
设备拓扑图：



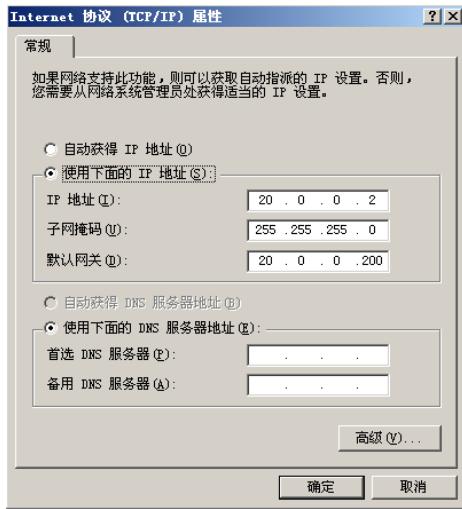
1.3.4 实验步骤

1.3.4.1 配置虚拟机 IP 地址

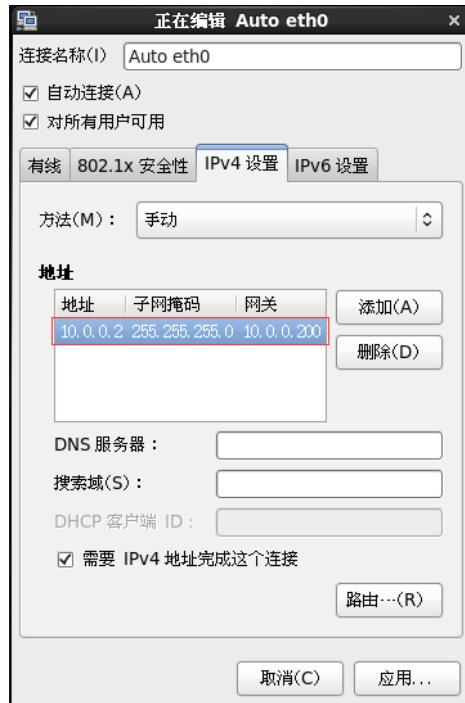
将操作机 Windows 7 的 IP 地址设置为 192.168.1.2，子网掩码设为 255.255.255.0，默
认网关设为 192.168.1.200。



将 Windows server 2003 的 IP 地址设为 20.0.0.2，子网掩码设为 255.255.255.0， 默认网关设为 20.0.0.200。



将服务器 CentOS 6.5 的 IP 地址设置为 10.0.0.2，子网掩码设为 255.255.255.0，网关设为 10.0.0.200。



作为防火墙的另一台 CentOS 6.5 安装了 3 块网卡，将网卡 eth0 的 IP 地址设置为 192.168.1.200，将网卡 eth1 的 IP 地址设置为 10.0.0.200，将网卡 eth2 的 IP 地址设置为 20.0.0.200，3 块网卡的子网掩码都设置为 255.255.255.0。

```

root@localhost:~/桌面
[root@localhost 桌面]# ifconfig
eth0      Link encap:Ethernet HWaddr 00:0c:29:fc:ab:5f
          inet addr: 192.168.1.200 Bcast: 192.168.1.255 Mask: 255.255.255.0
          inet6 addr: fe80::20c:29ff:feab:5f/64 Scope: Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets: 32832 errors:0 dropped:0 overruns:0 frame:0
          TX packets: 63 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes: 2611433 (2.4 MB) TX bytes: 3058 (2.9 kB)
          Interrupt: 18 Base address: 0x2024

eth1      Link encap:Ethernet HWaddr 00:0c:29:fc:ab:49
          inet addr: 10.0.0.200 Bcast: 10.0.0.255 Mask: 255.255.255.0
          inet6 addr: fe80::20c:29ff:feab:49/64 Scope: Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets: 32832 errors:0 dropped:0 overruns:0 frame:0
          TX packets: 12 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes: 2054453 (1.9 MB) TX bytes: 816 (816.0 b)
          Interrupt: 19 Base address: 0x2044

eth2      Link encap:Ethernet HWaddr 00:0c:29:fc:ab:83
          inet addr: 20.0.0.200 Bcast: 20.0.0.255 Mask: 255.255.255.0
          inet6 addr: fe80::20c:29ff:feab:83/64 Scope: Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets: 26181 errors:0 dropped:0 overruns:0 frame:0
          TX packets: 21 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes: 2026770 (1.9 MB) TX bytes: 1184 (1.1 kB)
          Interrupt: 18 Base address: 0x4242

lo       Link encap:Local Loopback
          inet addr: 127.0.0.1 Mask: 255.0.0.0
          inet6 addr: ::1/128 Scope: Host
          UP LOOPBACK RUNNING MTU:6436 Metric:1
          RX packets: 128 errors:0 dropped:0 overruns:0 frame:0
          TX packets: 128 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes: 9052 (9.7 kB) TX bytes: 9052 (9.7 kB)

[root@localhost 桌面]#

```

开启网卡之间的 IP 转发功能，在终端输入命令

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```

再输入命令

```
cat /proc/sys/net/ipv4/ip_forward
```

可以看到 /proc/sys/net/ipv4/ip_forward 中数值为 1，说明 IP 转发功能已经打开。

```

root@localhost:~/桌面
[root@localhost 桌面]# echo "1" > /proc/sys/net/ipv4/ip_forward
[root@localhost 桌面]# cat /proc/sys/net/ipv4/ip_forward
1
[root@localhost 桌面]#

```

要想永久使用 IP 转发，需要修改 /etc/sysctl.conf 文件，修改下面一行的值：

```
net.ipv4.ip_forward=0
```

将 0 改为 1。修改后可以重启系统来使修改生效，也可以执行命令

```
sysctl -p /etc/sysctl.conf
```

来使修改生效。

1.3.4.2 测试 win7、2003 和服务器 centos 的是否连通

在操作机 Windows 7 上打开 cmd，分别输入命令“ping 10.0.0.2”和“ping 20.0.0.2”，都可以 ping 通。

```

Administrator: Windows 命令处理器
Microsoft Windows (版本 6.1.7601)
版权所有 © 2009 Microsoft Corporation。保留所有权利。
C:\Windows\System32>ping 192.168.1.2
正在 Ping 192.168.1.2 具有 32 字节的数据:
来自 192.168.1.2 的回复: 字节=32 时间<1ms TTL=64

192.168.1.2 的 Ping 统计信息:
    软件包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
往返行程的估计时间(以毫秒为单位):
    最短 = 0ms, 最长 = 0ms, 平均 = 0ms

C:\Windows\System32>ping 20.0.0.2
正在 Ping 20.0.0.2 具有 32 字节的数据:
来自 20.0.0.2 的回复: 字节=32 时间<1ms TTL=127

20.0.0.2 的 Ping 统计信息:
    软件包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
往返行程的估计时间(以毫秒为单位):
    最短 = 0ms, 最长 = 1ms, 平均 = 0ms

C:\Windows\System32>

```

在 Windows server 2003 上打开 cmd, 分别输入命令“ping 192.168.1.2”和“ping 10.0.0.2”, 都可以 ping 通。

```

Administrator: Windows 命令提示符
Microsoft Windows (版本 5.2.3790)
(C) 版权所有 1985-2003 Microsoft Corp.

C:\Documents and Settings\Administrator.MHS-06207C07D97>ping 192.168.1.2
Pinging 192.168.1.2 with 32 bytes of data:
Reply from 192.168.1.2: bytes=32 time<1ms TTL=64
Reply from 192.168.1.2: bytes=32 time<1ms TTL=64
Reply from 192.168.1.2: bytes=32 time=2ms TTL=64
Reply from 192.168.1.2: bytes=32 time=<1ms TTL=64

Ping statistics for 192.168.1.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    约定往返时间在毫秒内:
        Minimum = 0ms, Maximum = 2ms, Average = 0ms

C:\Documents and Settings\Administrator.MHS-06207C07D97>ping 10.0.0.2
Pinging 10.0.0.2 with 32 bytes of data:
Reply from 10.0.0.2: bytes=32 time<1ms TTL=64
Reply from 10.0.0.2: bytes=32 time=2ms TTL=64
Reply from 10.0.0.2: bytes=32 time<1ms TTL=64
Reply from 10.0.0.2: bytes=32 time<1ms TTL=64

Ping statistics for 10.0.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    约定往返时间在毫秒内:
        Minimum = 0ms, Maximum = 2ms, Average = 0ms

C:\Documents and Settings\Administrator.MHS-06207C07D97>

```

在服务器 CentOS 6.5 上打开终端, 分别输入命令“ping 192.168.1.2”和“ping 20.0.0.2”, 都可以 ping 通。

```

root@localhost:~桌面
文件(F) 编辑(E) 查看(V) 搜索(S) 帮助(H)
[root@localhost 桌面]# ping 192.168.1.2
PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data
64 bytes from 192.168.1.2: icmp_seq=1 ttl=128 time=0.466 ms
64 bytes from 192.168.1.2: icmp_seq=2 ttl=128 time=0.188 ms
64 bytes from 192.168.1.2: icmp_seq=3 ttl=128 time=0.448 ms
^C
--- 192.168.1.2 ping statistics ---
4 packets transmitted, 3 received, 25% packet loss, time 3976ms
rtt min/avg/max/mdev = 0.187/0.466/0.127 ms
[root@localhost 桌面]# ping 20.0.0.2
PING 20.0.0.2 (20.0.0.2) 56(84) bytes of data.
64 bytes from 20.0.0.2: icmp_seq=1 ttl=127 time=2.80 ms
64 bytes from 20.0.0.2: icmp_seq=2 ttl=127 time=1.22 ms
64 bytes from 20.0.0.2: icmp_seq=3 ttl=127 time=0.851 ms
64 bytes from 20.0.0.2: icmp_seq=4 ttl=127 time=1.84 ms
^C
--- 20.0.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3087ms
rtt min/avg/max/mdev = 0.851/1.680/2.803/0.738 ms
[root@localhost 桌面]#

```

至此, 成功搭建完 iptables 的实验环境。

1.3.5 实验目的-了解 iptables 中常见的一些相关名词和术语

了解 iptables 中常见的一些相关名词和术语。

1.3.6 实验原理

- iptables 是与 Linux 内核集成的 IP 信息包过滤系统。如果 Linux 系统连接到因特网或 LAN、服务器或连接 LAN 和因特网的代理服务器，则该系统有利于在 Linux 系统上更好地控制 IP 信息包过滤和防火墙配置。
- 防火墙在做信息包过滤决定时，有一套遵循和组成的规则，这些规则存储在专用的信息包过滤表中，而这些表集成在 Linux 内核中。在信息包过滤表中，规则被分组放在我们所谓的链（chain）中。而“netfilter/iptables”IP 信息包过滤系统是一款功能强大的工具，可用于添加、编辑和移除规则。

1.3.6.1 iptables 防火墙

- Netfilter/Iptables（以下简称 iptables）是 Unix/Linux 自带的一款优秀且开放源码的完全基于包过滤的防火墙工具，它的功能十分强大，使用非常灵活，可以对流入和流出服务器的数据包进行很精细的控制。特别是它可以在一台非常低的硬件配置下跑得非常好。
- Netfilter/Iptables 的最大优点是它可以配置有状态的防火墙，这是老一辈 ipfwadm 和 ipchains 等以前的工具都无法提供的一种重要功能。Iptables 主要工作在 OSI 的二、三、四层，即数据链路层、网络层和传输层，如果重新编译内核，也可以支持 7 层控制。
- Iptables 防护墙由两个组件 netfilter 和 iptables 组成。
- netfilter 组件也称为内核空间（kernel space），是内核的一部分，由一些信息包过滤表组成，这些表包含内核用来控制信息包过滤处理的规则集。
- iptables 组件是一种工具，也称为用户空间（user space），它使插入、修改和除去信息包过滤表中的规则变得容易。

1.3.6.2 iptables 名词和术语

- 容器：在 iptables 中，容器用来描述包含或者属于的关系。
- Netfilter：Netfilter 是表（tables）的容器。
- 表（tables）：表（tables）是链的容器，即所有的链（chains）都属于对应的表（tables）。
- 链（chains）：链（chains）是规则（policy）的容器。
- 规则（policy）：规则（policy）是 iptables 一系列过滤信息的规范和具体方法。

我们可以用一个通俗易懂的例子表现这几个术语之间的关系。

Netfilter	tables(表)	chains(链)	policy(规则)
一栋楼	楼里的房间	房间里的柜子	柜子里的衣服摆放规则

1.3.7 实验目的-了解 iptables 中的表和链

了解 iptables 中的表和链。

1.3.8 实验原理

- Netfilter/iptables 是表 (tables) 的容器, iptables 包含 4 个表, 分别是 filter、nat、mangle 和 raw。
- iptables 的表 (tables) 是链 (chains) 的容器, iptables 包含 5 个链, 分别是 INPUT、OUTPUT、FORWARD、PREROUTING 和 POSTROUTING。

1.3.9 实验原理

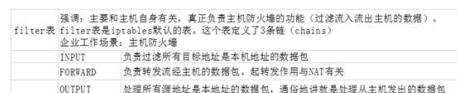
1.3.9.1 iptables 的表 (tables) 和链 (chains)

虽然 iptables 包含 4 个表, 但是因为表 raw 在实际中很少或者基本上用不到, 所以默认情况下, iptables 根据功能和表的定义划分包含 3 个表: filter、nat 和 mangle, 其中, 比较常用到的是 filter 和 nat。每个表又包含不同的操作链 (chains)。

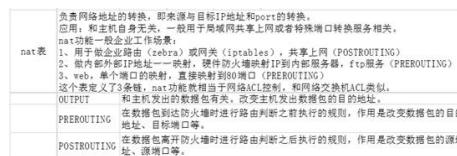
下面的表格展示了表和链的对应关系, 其中, 1 表示有, 0 则表示没有。注意, 所有链名必须大写。

表 (tables)/链 (chains)	INPUT	FORWARD	OUTPUT	PREROUTING	POSTROUTING
filter	1	1	1	0	0
nat	0	0	1	1	1
mangle	1	1	1	1	1

下图展示了 filter 表和链 INPUT、FORWARD 和 OUTPUT 之间的对应关系。



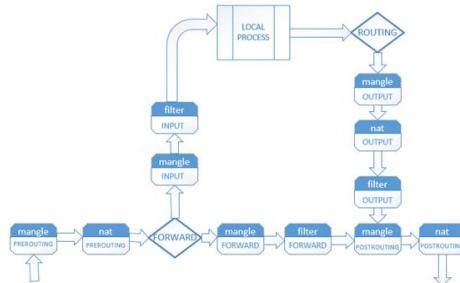
下图展示了 nat 表和链 OUTPUT、PREROUTING 和 POSTROUTING 之间的对应关系。



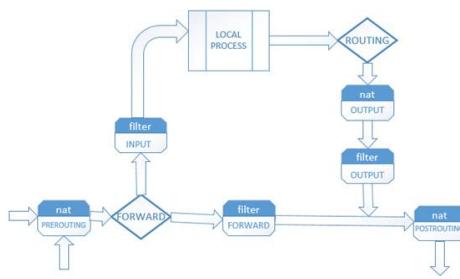
Mangle 表主要负责修改数据包特殊的路由标记, 如 TTL、TOS、MARK 等。由于 mangle 表与特殊标记相关, 一般情况下用不到这个表, 这里就不做详细介绍了。

1.3.9.2 iptables 表和链工作的流程图

下图清晰的描绘了 netfilter 对包的处理流程。



为了更好的学习 iptables，可以将上图进行简化。



上图可分为两个流程，上面的流程主要是 NAT 功能，在企业中常用于局域网共享、将外部 IP 和端口映射为内部 IP 和端口等。下面的流程主要是 FILTER 功能，即防火墙功能，企业中主要的应用是服务器防火墙。

1.3.10 实验目的-了解 iptables 常用的基本命令

了解 iptables 常用的基本命令.

1.3.11 实验原理

iptables 是用来设置、维护和检查 Linux 内核的 IP 包过滤规则的。可以定义不同的表，每个表都包含几个内部的链，也能包含用户自定义的链。每个链都是一个规则列表，对对应的包进行匹配：每条规则指定应当如何处理与之相匹配的包。这被称作 target (目标)，也可以跳向同一个表内的用户定义的链。

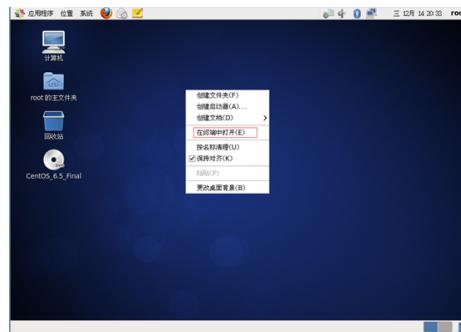
1.3.12 实验环境

操作系统：CentOS 6.5

1.3.13 实验步骤

1.3.13.1 查看 iptables 使用方法

一般而言，我们学习某个软件的使用方法，除了在网上或者使用书籍进行学习，最快捷有效的方式就是直接查看帮助。右键单击桌面，选择“在终端中打开”。



在终端中输入命令

```
iptables -h
```

则会列出 iptables 的使用方法。若想要看 iptables 更加详细的使用方法，可以使用命令

```
man iptables
```

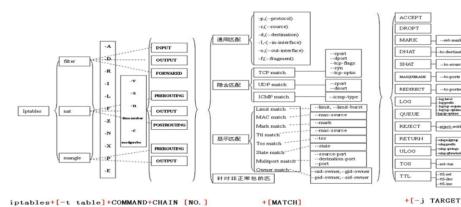
1.3.13.2 iptables 基本命令

iptables 命令的通用格式为：

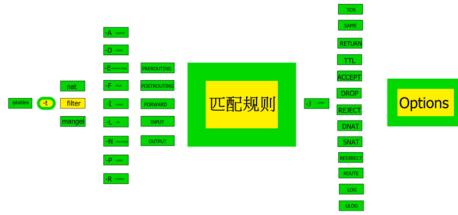
```
iptables [-t tables] COMMAND chain [-m matchname [per-match-options]] -j
        ↳ targetname [per-target-options]
```

其中，tables 包括 filter、nat、mangle 和 raw，chain 包括 INPUT、OUTPUT、FORWARD、PREROUTING 和 POSTROUTING。当不指定表时，默认表为 filter。

iptables 命令的使用格式可用图表示如下。



对上图进行简化。



下面对 COMMAND 进行介绍。

1. 常见用于链管理的参数有：

- -N: new, 自定义一条新的规则链
- -X: delete/drop, 删自定义的规则链, 指明名字删除一个, 不指明将删除多个
- -P: policy, 设置默认策略, 指明是黑名单还是白名单, 对于 filter 表中的链而言, 其默认策略有 ACCEPT (接受)、DROP (丢弃) 和 REJECT (拒绝)
- -E: 重命名自定义链, 引用计数不为 0 的自定义链, 不能被重命名或者删除

2. 常见的用于规则管理的参数有：

- -A: append, 追加规则
- -I: insert, 插入规则, 需要指明位置, 省略时表示第一条
- -D: delete, 删除规则, 指明规则号删除或者指明规则本身
- -R: replace, 替换指定链上的指定规则, 指明替换号或者指明替换规则本身
- -F: flush, 清空指定的规则链, 指明链, 即清空指定链上的规则; 不指明链, 即清空所有链上的规则
- -Z: zero, 置零, 匹配到的报文个数或者匹配到的所有报文的大小之和 (字节数)

3. 查看信息的常用参数有：

- -L: 必选, list, 列出指定链上的所有规则, 下面的几个为附加子参数
- -n: numeric, 以数据格式显示地址和端口
- -v: verbose, 详细信息, vv 或者 vvv (v 的个数越多越详细)
- -x: exactly(精确的), 显示计数器结果的精确值
- --line-numbers: 显示规则的序号

下面对匹配条件进行介绍。

1. 基本匹配条件：无需加载任何模块，由 iptables/netfilter 自行提供

- -s, --source address[/mask] [, ...]: 检查报文中源 IP 地址, 是否符合此处指定的地址或范围

- `-d,--destination address[/mask] [, ...]`: 检查报文中的目标 IP 地址, 是否符合此处指定的地址或范围
 - `-p,--protocol protocol`: 表明在传输层应用的协议, 其可以有 `tcp`, `udp`, `udplite`, `icmp`,`icmpv6`,`esp`,`ah`,`sctp`,`mh` 或者 `all`(包括 `tcp`、`udp` 和 `icmp` 3 个)
 - `-i,--in-interface name`: 只能应用于数据报文流入的接口, 作用于链 INPUT, FORWARD 和 PREROUTING
 - `-o,--out-interface name`: 只能应用于数据报文流出的接口, 作用于链 OUTPUT, FORWARD 和 POSTROUTING
2. 隐匿扩展: 不需要手动加载扩展模块, 因为它们是对协议的扩展, 所以但凡使用 `-p` 指明了协议, 就表示已经指明了要扩展的模块
- `tcp` 协议:
 - (a) `--source-port`,`--sport port[:port]`: 匹配报文的源端口, 可以是端口范围
 - (b) `--destination-port`,`--dport port[:port]`: 匹配报文的目标端口, 可以是端口范围
 - (c) `--tcp-flags mask comp`: `mask` 为必须要检查的标识位, 必须以逗号分隔; `comp` 必须为 1 的标识位, 必须以逗号分隔
 - (d) `--syn`: 匹配第一次握手
 - `udp` 协议:
 - (a) `--source-port`,`--sport port[:port]`: 匹配报文的源端口, 可以是端口范围
 - (b) `--destination-port`,`--dport port[:port]`: 匹配报文的目标端口, 可以是端口范围
 - `icmp` 协议: `--icmp {type[/code] | typename}`
3. 显示扩展: 必须使用 `-m` 选项手动加载模块, 其扩展模块路径为: `/lib64/xtables`, 其中大写的为目标扩展, 小写的为规则扩展。
4. `multiport` 扩展: 以离散方式定义多端口匹配, 但最多指定 15 个端口
- (a) `--source-ports`,`--sports port[,port|,port:port]`: 指定多个源端口
 - (b) `--destination-ports`,`--dports port[,port|,port:port]`: 指定多个源端口
 - (c) `--ports port[,port|,port:port]`: 指定多个目标及源端口
5. `iprange` 扩展: 指明连续的 (不过一般不能是整个网络)ip 地址范围
- (a) `--src-range from[-to]`: 源 IP 地址
 - (b) `--dst-range from[-to]`: 目标 IP 地址
6. `connlimit` 扩展: 对每客户端 IP 做并发连接数量匹配
- (a) `--connlimit-upto n`: 当现在的连接数量低于或等于这个数量 (n), 就匹配

(b) `--connlimit-above n`: 当现有的连接数量大于这个数量, 就匹配

7. limit 扩展: 基于收发报文的速率做匹配

(a) `--limit rate [/second|/minute|/hour]`: 平均速率

(b) `--limit-burst NUMBER`: 峰值数量, 默认 5 个

8. state 扩展: 根据连接追踪机制, 查检连接的状态, 跟 TCP 没有关系, 是内核中 netfilter 实现, 能实现 tcp, udp, icmp 的连接追踪, 内核会记录每一个连接 (放置在内存中), 谁通过什么协议访问什么服务, 访问的时间, 这种机制被称之为 conntrack 机制。也正是有了 state 扩展, iptables 成为了有连接追踪的防火墙, 安全性更高。是由 state 扩展提供, 库文件为 `ibxt_conntrack.so`。追踪连接功能在内核的内存空间中, 把出去和进来的连接通过模板建立关联关系. 追踪本机的请求和响应之间的关系, 状态如下几种:

(a) NEW: 新发起的请求

(b) ESTABLISHED: new 状态之后, 连接追踪模板中为其建立的条目失效之前期间内所有的通信状态

(c) RELATED: 相关的连接, 如 FTP 协议中的命令连接与数据连接之间的关系

(d) INVALID: 无效的连接, 如 tcp 状态全为 1 或者全为 0 的连接

(e) UNTRACKED: 未进行追踪的连接

调整连接追踪功能所容纳的最大连接数量:

```
/proc/sys/net/nf_conntrack_max
```

已经追踪到的并记录下来的连接:

```
/proc/net/nf_conntrack
```

不同协议的连接追踪状态时长 (可修改):

```
/proc/sys/net/netfilter
```

`--state STATE`: 多个 state 可以使用逗号分隔. iptables 的连接追踪表最大容量为

```
cat /proc/sys/ipv4/ip_conntrack_max
```

链接达到各种状态的超时后, 会从表中删除, 当模板满载时, 后续的链接可能会超时, 可以有如下两种解决方法 (但加大 max 值, 也会加大内存的压力)

(a) 修改 max 的内核参数:

```
vim /etc/sysctl.conf
```

(b) 降低 nf_conntrack timeout 时间:

```
vim /etc/sysctl.conf
```

放行被动模式的 ftp 服务，需手动加载 nf_conntrack_ftp:

```
modprobe nf_conntrack_ftp
```

target 的分类:

1. ACCEPT: 接受
2. DROP: 丢弃
3. REJECT: 拒绝
4. RETURN: 返回调用链
5. REDIRECT: 端口重定向
6. LOG: 记录日志, --log-level LEVEL 用于设置日志的等级, --log-prefix PFREFIX 用于设置日志的提示语句的前缀
7. MASK: 做防火墙标记
8. DNAT: 目标地址转换
9. SNAT: 源地址转换
10. MASQUERADE: 地址伪装

1.3.14 实验目的-熟悉 iptables 的表链操作

熟悉 iptables 的表链操作.

1.3.15 实验原理

通过 iptables 命令对防火墙 iptables 的表链进行相关操作（因环境策略不同，表链会有所不同），如查看、添加、删除、修改等等。

1.3.16 实验环境

操作系统: CentOS 6.5

1.3.17 实验步骤

1.3.17.1 Iptables 链操作

查看 CentOS 6.5 中防火墙 iptables 的表名（因环境策略不同，表会有所不同）。在终端输入命令

```
cat /proc/net/ip_tables_names
```

若无结果，请查看是否开启服务。可执行

```
service iptables restart
```

重启服务后再次执行，可以看到 iptables 包含 4 张表，分别是 mangle、raw、filter 和 nat（默认只有 filter 表，如果之前有对其他表进行过相关操作，才会有其他表）。

```
mangle
raw
filter
nat
```

在终端输入命令

```
iptables -t filter -L
```

查看表 filter 中的链及其链中的规则。`-t filter` 指定表 filter，也可不加，不加时默认表 filter。可知表 filter 有 3 条链：INPUT、FORWARD 和 OUTPUT，这 3 条链的默认规则都为 ACCEPT。

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

若要查看某条链上的规则，在 `-L` 参数后指明链名。例如在终端输入命令

```
iptables -L FORWARD
```

查看表 filter 中链 FORWARD 上的规则。

```
[root@localhost ~]# iptables -L FORWARD
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
[root@localhost ~]#
```

在终端输入命令

```
iptables -t nat -L
```

查看表 nat 中的链及其链中的规则。可知表 nat 有 3 条链：PREROUTING、POSTROUTING 和 OUTPUT，这 3 条链的默认规则都为 ACCEPT。

```
[root@localhost ~]# iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
[root@localhost ~]#
```

在终端输入命令

```
iptables -t mangle -L
```

查看表 mangle 中的链及其链中的规则。可知表 nat 有 5 条链：PREROUTING、INPUT、FORWARD、OUTPUT 和 POSTROUTING，这 5 条链的默认规则都为 ACCEPT。

```
[root@localhost ~]# iptables -t mangle -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
[root@localhost ~]#
```

在终端输入命令

```
iptables -t raw -L
```

查看表 nat 中的链及其链中的规则。可知表 raw 有 2 条链：PREROUTING 和 OUTPUT，这 2 条链的默认规则都为 ACCEPT。

```
[root@localhost ~]# iptables -t raw -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
[root@localhost ~]#
```

在终端输入命令

```
cat /etc/sysconfig/iptables
```

(因环境策略不同，表中链及链中规则会有所不同)，查看防火墙配置文件，可以查看到 iptables 所有表中的链及其链中规则（默认只有 filter 表，如果之前有对其他表进行过相关操作，才会有其他表）。

```
[root@localhost 桌面]# cat /etc/sysconfig/iptables
# Generated by iptables-save v1.4.7 on Tue Dec 20 14:37:24 2016
*mangle
:PREROUTING ACCEPT [7601:731006]
:INPUT ACCEPT [58:4800]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [26:2072]
COMMIT
# Completed on Tue Dec 20 14:37:24 2016
# Generated by iptables-save v1.4.7 on Tue Dec 20 14:37:24 2016
*filter
:PREROUTING ACCEPT [7610:732242]
:OUTPUT ACCEPT [26:2072]
COMMIT
# Completed on Tue Dec 20 14:37:24 2016
# Generated by iptables-save v1.4.7 on Tue Dec 20 14:37:24 2016
:filter
:INPUT ACCEPT [2:473]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -p tcp --dport 80 -j DROP
COMMIT
# Completed on Tue Dec 20 14:37:24 2016
# Generated by iptables-save v1.4.7 on Tue Dec 20 14:37:24 2016
*nat
:PREROUTING ACCEPT [8993:857994]
:POSTROUTING ACCEPT [43:2878]
:OUTPUT ACCEPT [43:2878]
COMMIT
# Completed on Tue Dec 20 14:37:24 2016
[root@localhost 桌面]#
```

这 4 张表 5 条链中的规则 target 为目标；prot 为协议；opt 为 option，选项；source 为源地址，destination 为目的地地址。在终端输入命令

```
cat /proc/net/ip_tables_targets
```

查看 iptables 中的 targets。

```
[root@localhost 桌面]# cat /proc/net/ip_tables_targets
LOG
DNAT
SNAT
ERROR
[root@localhost 桌面]#
```

在终端输入命令

```
iptables -N simpleware1
```

在表 filter 中添加一条名为 simpleware1 的新链。这里没有使用 -t 指定表，默认是 filter 表，还可用 -t nat/mangle/raw 在指定的表中添加新的链。再使用

```
iptables -L
```

进行查看。

```
[root@localhost ~]# iptables -N simpleware1
[root@localhost ~]# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP      tcp  --  anywhere             anywhere            tcp dpt: http
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
Chain simpleware1 (0 references)
target     prot opt source               destination
[root@localhost ~]#
```

使用上一步的方法继续在表 filter 中添加两条新链 simpleware2 和 simpleware3。

```
[root@localhost ~]# iptables -N simpleware2
[root@localhost ~]# iptables -N simpleware3
[root@localhost ~]# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP      tcp  --  anywhere             anywhere            tcp dpt: http
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
Chain simpleware1 (0 references)
target     prot opt source               destination
Chain simpleware2 (0 references)
target     prot opt source               destination
Chain simpleware3 (0 references)
target     prot opt source               destination
[root@localhost ~]#
```

若想删除某条自定义的链，使用 **-X** 参数。在终端输入命令

```
iptables -X simpleware1
```

删除表 filter 中自定义的链 simpleware1。

```
[root@localhost ~]# iptables -X simpleware1
[root@localhost ~]# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP      tcp  --  anywhere             anywhere            tcp dpt: http
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
Chain simpleware2 (0 references)
target     prot opt source               destination
Chain simpleware3 (0 references)
[root@localhost ~]#
```

在终端输入命令

```
iptables -X
```

直接删除表 filter 中所有自定义的链。

```
[root@localhost ~]# iptables -X
[root@localhost ~]# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
[root@localhost ~]#
```

可以使用 -P 参数修改某条链的默认规则。例如在终端输入命令

```
iptables -P INPUT DROP
```

将表 filter 中的链 INPUT 的默认规则改为 DROP。

```
[root@localhost 截图]# iptables -P INPUT DROP
[root@localhost 截图]# iptables -L INPUT
Chain INPUT (policy DROP)
target     prot opt source               destination
DROP      tcp   --  anywhere             anywhere            tcp dpt:http
[root@localhost 截图]#
```

在终端输入如下命令，分别在 filter 表的 3 条链上添加一条防火墙规则。

```
[root@localhost 截图]# iptables -A INPUT -s 192.168.1.2 -j DROP
[root@localhost 截图]# iptables -A FORWARD -s 192.168.1.2 -j DROP
[root@localhost 截图]# iptables -A OUTPUT -s 192.168.1.2 -j DROP
[root@localhost 截图]# iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination
DROP      all   --  192.168.1.2          anywhere
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
DROP      all   --  192.168.1.2          anywhere
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
DROP      all   --  192.168.1.2          anywhere
[root@localhost 截图]#
```

在终端输入命令

```
iptables -F INPUT
```

清空 filter 表中 INPUT 链上的所有规则。

```
[root@localhost 截图]# iptables -F INPUT
[root@localhost 截图]# iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
DROP      all   --  192.168.1.2          anywhere
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
DROP      all   --  192.168.1.2          anywhere
[root@localhost 截图]#
```

在终端输入命令

```
iptables -F
```

不指明链名时，删除表 filter 中所有链上的规则。

```
[root@localhost 截图]# iptables -F
[root@localhost 截图]# iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
[root@localhost 截图]#
```

1.3.18 实验目的-配置 iptables 禁止访问 ftp 服务

通过配置 iptables 防火墙禁止访问 ftp 服务

1.3.19 实验原理

1. Iptables 防火墙在做信息包过滤决定时，有一套遵循和组成的规则，这些规则存储在专用的信息包过滤表中，而这些表集成在 Linux 内核中。在信息包过滤表中，规则被分组放在我们所谓的链（chain）中。而 netfilter/iptables IP 信息包过滤系统是一款功能强大的工具，可用于添加、编辑和移除规则。
2. FTP 是 File Transfer Protocol（文件传输协议）的英文简称，而中文简称为“文传协议”。用于 Internet 上的控制文件的双向传输。同时，它也是一个应用程序（Application）。基于不同的操作系统有不同的 FTP 应用程序，而所有这些应用程序都遵守同一种协议以传输文件。在 FTP 的使用当中，用户经常遇到两个概念：“下载”（Download）和“上传”（Upload）。“下载”文件就是从远程主机拷贝文件至自己的计算机上；“上传”文件就是将文件从自己的计算机中拷贝至远程主机上。用 Internet 语言来说，用户可通过客户机程序向（从）远程主机上传（下载）文件。

1.3.20 实验环境

操作系统：

- Windows 7
- Windows server 2003
- CentOS 6.5
- CentOS 6.5

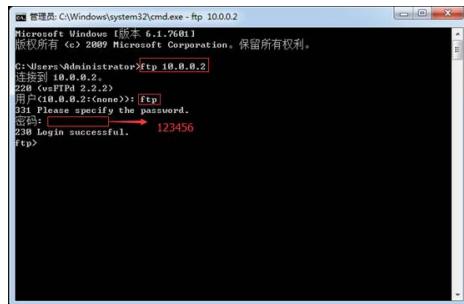
1.3.21 实验步骤

1.3.21.1 在当前环境下内外网均可连接 ftp 服务

打开 Windows 7 的 cmd，输入

```
ftp 10.0.0.2
```

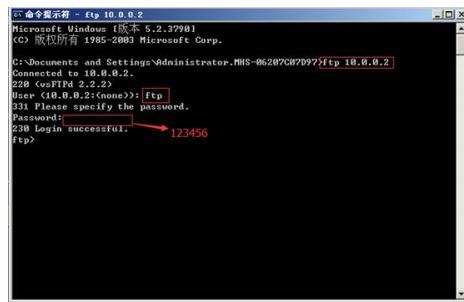
输入用户名 ftp，密码 123456，登录成功。



打开 Windows server 2003 的 cmd，输入

```
ftp 10.0.0.2
```

输入用户名 ftp，密码 123456，登录成功。



1.3.21.2 配置 iptables，禁止内外网访问 ftp 服务

在防火墙 CentOS 6.5 的终端输入如下命令，清空防火墙规则。

```
iptables -F
iptables -X
iptables -Z
iptables -L -n --line-numbers
```



配置防火墙，禁止内外网访问 ftp 服务。因为这里我们使用 CentOS 6.5 充当防火墙，数据会在网卡之间进行转发，所以我们这里对 filter 表的链 FORWARD 进行配置。在终端输入如下命令，在转发的数据包中，如果目标端口为 21，则将该数据包丢弃。

```
iptables -t filter -A FORWARD -p tcp --dport 21 -j DROP
```

```
root@localhost:~/桌面
[root@localhost 桌面]# iptables -t filter -A FORWARD -p tcp --dport 21 -j DROP
[root@localhost 桌面]# iptables -L -n --line-numbers
Chain INPUT (policy ACCEPT)
num  target     prot opt source               destination
Chain FORWARD (policy ACCEPT)
num  target     prot opt source               destination
  1  DROP       tcp  --  0.0.0.0/0            0.0.0.0/0          tcp dpt:21
Chain OUTPUT (policy ACCEPT)
num  target     prot opt source               destination
[root@localhost 桌面]#
```

在终端输入如下命令，将添加的规则保存到 iptables 配置文件中，重启防火墙，查看防火墙规则。

```
root@localhost:~/桌面
[root@localhost 桌面]# /etc/init.d/iptables save
[root@localhost 桌面]# /etc/init.d/iptables restart
[root@localhost 桌面]# cat /etc/sysconfig/iptables
# Generated by iptables-save v1.4.7 on Thu Mar 9 09:44:09 2017
*nat
:INPUT ACCEPT [5 680]
:FORWARD ACCEPT [0 0]
:OUTPUT ACCEPT [1 0]
-A FORWARD -p tcp --dport 21 -j DROP
COMMIT
# Completed on Thu Mar 9 09:44:09 2017
# Generated by iptables-save v1.4.7 on Thu Mar 9 09:44:09 2017
*nat
:PREROUTING ACCEPT [1020:104217]
:POSTROUTING ACCEPT [26:1722]
:OUTPUT ACCEPT [24:1614]
COMMIT
# Completed on Thu Mar 9 09:44:09 2017
[root@localhost 桌面]#
```

此时在 Windows 7 的 cmd 上，输入

```
ftp 10.0.0.2
```

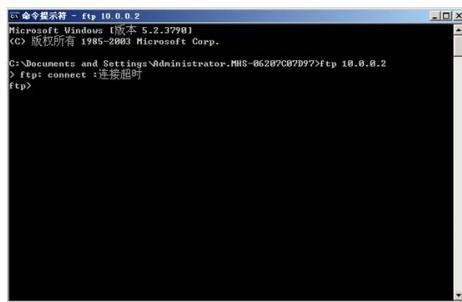
已经不能连接 ftp 服务了。

```
Administrator: C:\Windows\system32\cmd.exe - ftp 10.0.0.2
Microsoft Windows (版本 6.1.7601)
版权所有 © 2009 Microsoft Corporation。保留所有权利。
C:\Users\Administrator>ftp 10.0.0.2
ftp>
```

同样在 Windows server 2003 的 cmd 上，输入

```
ftp 10.0.0.2
```

也已经不能连接 ftp 服务了。



1.3.22 实验目的-配置 iptables 按网段访问 ftp 服务

通过配置 iptables 防火墙来按网段访问 ftp 服务.

1.3.23 实验原理

Iptables 防火墙在做信息包过滤决定时，有一套遵循和组成的规则，这些规则存储在专用的信息包过滤表中，而这些表集成在 Linux 内核中。在信息包过滤表中，规则被分组放在我们所谓的链（chain）中。而 netfilter/iptables IP 信息包过滤系统是一款功能强大的工具，可用于添加、编辑和移除规则。

1.3.24 实验环境

操作系统：

- Windows 7
- Windows server 2003
- CentOS 6.5
- CentOS 6.5

1.3.25 实验步骤

1.3.25.1 在当前环境下内外网均可连接 ftp 服务

打开 Windows 7 的 cmd，输入

```
ftp 10.0.0.2
```

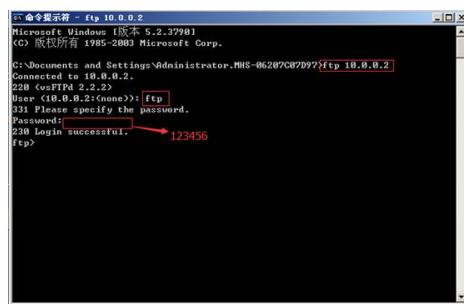
输入用户名 ftp，密码 123456，登录成功。



打开 Windows server 2003 的 cmd，输入

```
ftp 10.0.0.2
```

输入用户名 ftp，密码 123456，登录成功。



1.3.25.2 配置 iptables，按网段访问 ftp 服务

这里以 Windows server 2003 能够访问 ftp 服务，而 Windows 7 不能访问 ftp 服务为例。

在防火墙 CentOS 6.5 的终端输入如下命令，清空防火墙规则。

```
iptables -F  
iptables -X  
iptables -Z  
iptables -L -n --line-numbers
```

```
[root@localhost ~]# iptables -F
[root@localhost ~]# iptables -X
[root@localhost ~]# iptables -Z
[root@localhost ~]# iptables -L -n --line-numbers
Chain INPUT (policy ACCEPT)
num  target     prot opt source         destination
Chain FORWARD (policy ACCEPT)
num  target     prot opt source         destination
Chain OUTPUT (policy ACCEPT)
num  target     prot opt source         destination
[root@localhost ~]#
```

配置防火墙，按网段访问 ftp 服务。在终端输入如下命令，在转发的数据包中，如果源地址为 192.168.1.0/24，目标端口为 21，则将该数据包丢弃，同时允许源地址为 20.0.0.0/24，目标端口为 21 的数据包通过，这一条防火墙规则可加可不加。

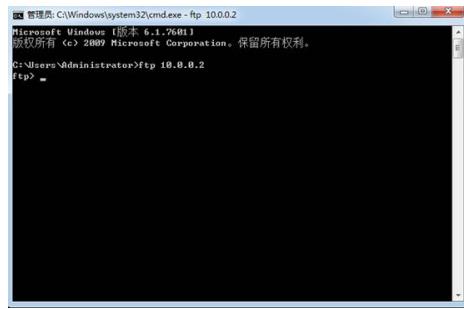
```
iptables -t filter -A FORWARD -s 192.168.1.0/24 -p tcp --dport 21 -j DROP
iptables -t filter -A FORWARD -s 20.0.0.0/24 -p tcp --dport 21 -j ACCEPT
/etc/init.d/iptables save
/etc/init.d/iptables restart
cat /etc/sysconfig/iptables
```

```
[root@localhost ~]# /etc/init.d/iptables save
iptables: 将防火墙规则保存到 /etc/sysconfig/iptables: [确定]
[root@localhost ~]# /etc/init.d/iptables restart
iptables: 读取防火墙规则: [确定]
iptables: 正在卸载模块: [确定]
iptables: 正在卸载模块: [确定]
iptables: 应用防火墙规则: [确定]
[root@localhost ~]# cat /etc/sysconfig/iptables
# Generated by iptables-save v1.4.7 on Thu Mar 9 10:44:32 2017
*xtables*
:PREROUTING ACCEPT [1053:94743]
:POSTROUTING ACCEPT [8:472]
:OUTPUT ACCEPT [4:272]
COMMIT
# Completed on Thu Mar 9 10:44:32 2017
# Generated by iptables-save v1.4.7 on Thu Mar 9 10:44:32 2017
*filter*
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A FORWARD -s 192.168.1.0/24 -p tcp -m tcp --dport 21 -j DROP
COMMIT
# Completed on Thu Mar 9 10:44:32 2017
[root@localhost ~]#
```

此时在 Windows 7 的 cmd 上，输入

```
ftp 10.0.0.2
```

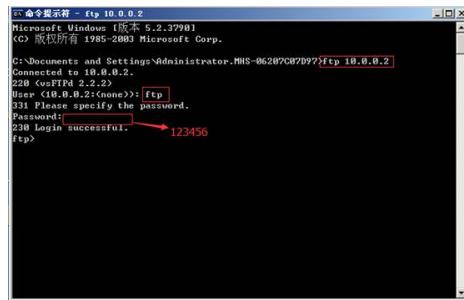
已经不能连接 ftp 服务了。



但在 Windows server 2003 的 cmd 上，输入

```
ftp 10.0.0.2
```

可以继续连接 ftp 服务。



1.3.26 实验目的-通过编写脚本配置防火墙

通过编写脚本配置防火墙.

1.3.27 实验原理

编写脚本，执行脚本，完成防火墙的配置.

1.3.28 实验环境

操作系统：CentOS 6.5

1.3.29 实验步骤

1.3.29.1 使用脚本配置防火墙

在防火墙 CentOS 6.5 桌面新建 `firewall.sh`，在终端输入命令

```
vi firewall.sh
```



编辑脚本 `firewall.sh`, 文件内容如下。

```
#!/bin/bash
IPT="/sbin/iptables"
#Remove any existing rules
$IPT -F
$IPT -X
$IPT -Z
#setting for loopback interface
$IPT -t filter -A INPUT -i lo -j ACCEPT
$IPT -t filter -A OUTPUT -o lo -j ACCEPT
#setting default firewall policy
$IPT -P OUTPUT ACCEPT
$IPT -P FORWARD DROP
$IPT -P INPUT DROP
#setting access rules
$IPT -t filter -A INPUT -s 192.168.1.0/24 -p all -j ACCEPT
#http
$IPT -t filter -A INPUT -p tcp --dport 80 -j ACCEPT
#icmp
$IPT -t filter -A INPUT -p icmp -m icmp --icmp-type any -j ACCEPT
#others RELATED
$IPT -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPT -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

```

root@localhost:~/桌面
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
#!/bin/bash

#IPT="/sbin/iptables"

#Remove any existing rules
#IPT -F
#IPT -X
#IPT -Z

#setting for loopback interface
#IPT -t filter -A INPUT -i lo -j ACCEPT
#IPT -t filter -A OUTPUT -o lo -j ACCEPT

#setting default firewall policy
#IPT -P OUTPUT ACCEPT
#IPT -P FORWARD DROP
#IPT -P INPUT DROP

#setting access rules
#IPT -t filter -A INPUT -s 192.168.10.0/24 -p all -j ACCEPT

#http
#IPT -t filter -A INPUT -p tcp --dport 80 -j ACCEPT

#icmp
#IPT -t filter -A INPUT -p icmp --icmp-type any -j ACCEPT

#other's RELATED
#IPT -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
#IPT -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

-- INSERT --

```

此时在终端输入命令

```
./firewall.sh
```

执行脚本时提示权限不够，输入命令

```
chmod +x firewall.sh
```

为其添加权限。再执行脚本，则可执行成功。

```

root@localhost:~/桌面
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
root@localhost 桌面]# ./firewall.sh
bash: ./firewall.sh: 权限不够
[root@localhost 桌面]# chmod +x firewall.sh
[root@localhost 桌面]# ./firewall.sh
[root@localhost 桌面]#

```

在终端输入命令

```
iptables -L -n --line-numbers
```

查看防火墙规则，可以看到防火墙规则添加成功。

```

root@localhost:~/桌面
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
[root@localhost 桌面]# iptables -L -n --line-numbers
Chain INPUT (policy ACCEPT)
num  target     prot opt source               destination
1    ACCEPT     all  --  0.0.0.0/0          0.0.0.0/0
2    ACCEPT     all  --  192.168.10.0/24    0.0.0.0/0
3    ACCEPT     tcp  --  0.0.0.0/0          0.0.0.0/0
4    ACCEPT     icmp --  0.0.0.0/0         0.0.0.0/0
5    ACCEPT     all  --  0.0.0.0/0          0.0.0.0/0
                                         state RELATED,ESTABLISHED

Chain FORWARD (policy DROP)
num  target     prot opt source               destination
Chain OUTPUT (policy ACCEPT)
num  target     prot opt source               destination
1    ACCEPT     all  --  0.0.0.0/0          0.0.0.0/0
2    ACCEPT     all  --  0.0.0.0/0          0.0.0.0/0
                                         state RELATED,ESTABLISHED
[root@localhost 桌面]#

```

但此时添加的防火墙规则只存在于内存中，并没有写入 iptables 配置文件中。一旦重启就会消失。将添加的防火墙规则永久保存在配置文件中，在终端输入命令

```
/etc/init.d/iptables save
```



在终端输入命令

```
cat /etc/sysconfig/iptables
```

查看 iptables 配置文件。



可以看到此时的防火墙与手动执行 iptables 命令配置防火墙实验中配置防火墙一样。相比于手动配置防火墙，使用脚本配置防火墙更加方便和快捷，不用一条一条手动地添加规则。

1.3.30 实验目的-实现外部 IP 地址映射到服务器

使用 iptables 命令配置防火墙，实现外部 IP 地址映射到服务器。

1.3.31 实验原理

通过配置 NAT 表实现外部 IP 地址映射到服务器。

1.3.32 实验环境

操作系统：

- Windows 7
- Windows server 2003
- CentOS 6.5

1.3.33 实验步骤

1.3.33.1 将外部 IP 端口地址映射到内部服务器

打开防火墙 CentOS 6.5 的终端，输入命令

```
lsof -i :80
```

若无 http 服务，输入

```
/etc/init.d/httpd start
```

开启 http 服务，再使用

```
lsof -i :80
```

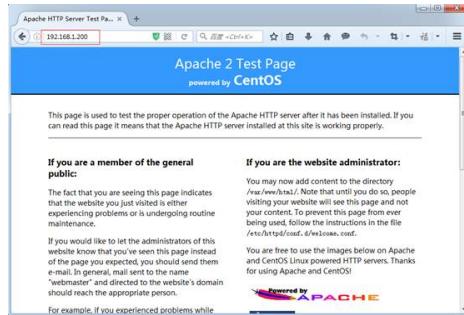
进行查看时，可以看到 http 服务已成功开启。

```
root@localhost:~# lsof -i :80
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
httpd 6772 root 4u IPv6 117251 0t0 TCP *:http (LISTEN)
httpd 6774 apache 4u IPv6 117251 0t0 TCP *:http (LISTEN)
httpd 6775 apache 4u IPv6 117251 0t0 TCP *:http (LISTEN)
httpd 6776 apache 4u IPv6 117251 0t0 TCP *:http (LISTEN)
httpd 6777 apache 4u IPv6 117251 0t0 TCP *:http (LISTEN)
httpd 6778 apache 4u IPv6 117251 0t0 TCP *:http (LISTEN)
httpd 6779 apache 4u IPv6 117251 0t0 TCP *:http (LISTEN)
httpd 6780 apache 4u IPv6 117251 0t0 TCP *:http (LISTEN)
httpd 6781 apache 4u IPv6 117251 0t0 TCP *:http (LISTEN)
httpd 6782 apache 4u IPv6 117251 0t0 TCP *:http (LISTEN)
[root@localhost:~#]
```

此时防火墙 nat 表无规则。

```
[root@localhost ~]# iptables -t nat -L -n --line-numbers
Chain PREROUTING (policy ACCEPT)
num target     prot opt source          destination
Chain POSTROUTING (policy ACCEPT)
num target     prot opt source          destination
Chain OUTPUT (policy ACCEPT)
num target     prot opt source          destination
[root@localhost ~]#
```

此时在 Windows 7 上打开火狐浏览器，在地址栏输入 `http://192.168.1.200`，打开的是防火墙 CentOS 6.5 上的 http 服务，即 Apache。



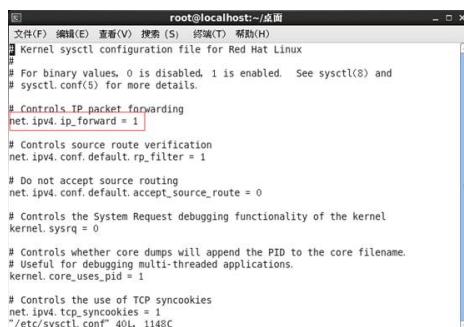
在防火墙 CentOS6.5 的终端输入命令

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

开启防火墙数据包转发功能。输入命令

```
cat /proc/sys/net/ipv4/ip_forward
```

进行查看。此外，必须保证 /etc/sysctl.conf 文件中 net.ipv4.ip_forward 的值为 1。



在防火墙 CentOS 6.5 的终端输入如下命令，

```
iptables -t nat -A PREROUTING -d 192.168.1.200 -p tcp --dport 80 -j DNAT
    ↳ --to-destination 20.0.0.2:80
iptables -t nat -A POSTROUTING -d 20.0.0.2 -p tcp --dport 80 -j SNAT
    ↳ --to-source 192.168.1.200
iptables -t nat -L -n --line-numbers
```

将外部 IP 地址端口 192.168.1.200:80 映射到内部服务器 20.0.0.2:80。

```
[root@localhost ~]# iptables -t nat -A PREROUTING -d 192.168.1.200 -p tcp --dport 80 -j DNAT --to-destination 20.0.0.2:80
[root@localhost ~]# iptables -t nat -A POSTROUTING -d 20.0.0.2 -p tcp --sport 80 -j SNAT --to-source 192.168.1.200
[root@localhost ~]# iptables -t nat -L -n --line-numbers
Chain PREROUTING (policy ACCEPT)
num  target     prot opt source         destination
1   DNAT       tcp  --  0.0.0.0/0      192.168.1.200    tcp dpt:80 to:20.0.0.2:80
Chain POSTROUTING (policy ACCEPT)
num  target     prot opt source         destination
1   SNAT       tcp  --  0.0.0.0/0      20.0.0.2        tcp dpt:80 to:192.168.1.200
Chain OUTPUT (policy ACCEPT)
num  target     prot opt source         destination
[root@localhost ~]#
```

在终端输入命令

```
/etc/init.d/iptables save
```

保存配置的防火墙规则。输入命令

```
cat /etc/sysconfig/iptables
```

查看配置的防火墙规则。

```
[root@localhost ~]# /etc/init.d/iptables save
iptables: 表 'filter' 的规则数已修改为 1000000。 [确定]
[root@localhost ~]# cat /etc/sysconfig/iptables
# Generated by iptables-save v1.4.7 on Tue Dec 20 19:40:58 2016
*nat
:PREROUTING ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-PREROUTING -d 192.168.1.200 -p tcp --dport 80 -j DNAT --to-destination 20.0.0.2:80
-A POSTROUTING -d 20.0.0.2/32 -p tcp -m tcp --sport 80 -j SNAT --to-source 192.168.1.200
COMMIT
# Generated on Tue Dec 20 19:40:58 2016
# Generated by iptables-save v1.4.7 on Tue Dec 20 19:40:58 2016
*filter
:INPUT ACCEPT [1:8 -i ens3]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
COMMIT
# Generated on Tue Dec 20 19:40:58 2016
[root@localhost ~]#
```

此时在操作机 Windows7 上访问 <http://192.168.1.200> 时访问的是内部服务器的网站 <http://20.0.0.2>。



1.4 数据库安全

1.4.1 实验目的-MSSQL 攻击日志分析

1. 破解 SQL Server 2008 的 sa 密码
2. 抓取破解过程中的数据包并分析
3. 查看并分析 SQL Server 日志文件

1.4.2 实验原理

TCP 协议与三次握手: TCP(Transmission Control Protocol: 传输控制协议) , TCP 是主机对主机层的传输控制协议, 提供可靠的连接服务, 采用三次握手确认建立一个连接。

位码即 tcp 标志位, 有 6 种标示:

- SYN(synchronous 建立联机)
- ACK(acknowledgement 确认)
- PSH(push 传送)
- FIN(finish 结束)
- RST(reset 重置)
- URG(urgent 紧急)

三次握手过程如下:

1. 第一次握手: 主机 A 发送位码为 $\text{SYN} = 1$, 随机产生 SEQ number 的数据包到服务器, 主机 B 由 $\text{SYN} = 1$ 知道, A 要求建立联机;
2. 第二次握手: 主机 B 收到请求后要确认联机信息, 向 A 发送 $\text{ACK number} = (\text{主机 A 的 SEQ} + 1), \text{SYN} = 1, \text{ACK} = 1$, 随机产生 SEQ number 的包;
3. 第三次握手: 主机 A 收到后检查 ACK number 是否正确, 即第一次发送的 $\text{SEQ number} + 1$, 以及位码 ack 是否为 1, 若正确, 主机 A 会再发送 $\text{ACK number} = (\text{主机 B 的 SEQ} + 1), \text{ACK} = 1$, 主机 B 收到后确认 SEQ 值与 $\text{ACK} = 1$ 则连接建立成功。

完成三次握手, 主机 A 与主机 B 开始传送数据。

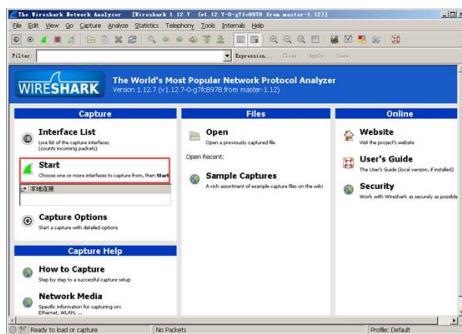
1.4.3 实验环境

- Kali Linux: 192.168.1.2 hydra
- Windows server 2003: 192.168.1.3 SQL Server 2008 R2

1.4.4 实验步骤

1.4.4.1 破解 sa 密码, 并用 wireshark 抓取破解数据包

在 Windows server 2003 上打开 wireshark, 并抓取数据包。



在 Kali 上，查看用户名文件 userlist.txt 和密码文件 passlist.txt。

```
root@localhost:~/桌面
文件(F) 编辑(E) 查看(V) 搜索(S) 标题(T) 帮助(H)
root@localhost:~/桌面# cd /root/桌面
root@localhost:~/桌面# cat userlist.txt
sa
admin
remote
john
user
root
simpleware
root
test
system
supervisor
administrator
simplexue
dts
xmu
root@localhost:~/桌面# cat passlist.txt
test
user
albatross
password
secret
123456
manager
12345678
qazwsx
qerty
asdfgh
Simplexue123
zxcbn
root@localhost:~/桌面#
```

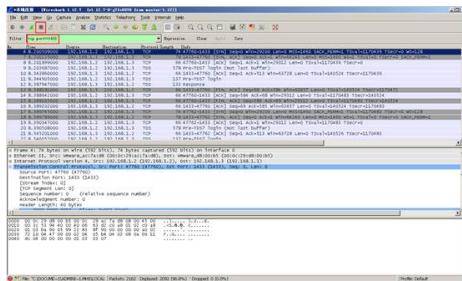
在终端输入命令

```
hydra -L userlist.txt -P passlist.txt 192.168.1.3 mssql -t 1
```

开始破解 mssql，可以看到破解出 sa 的密码为 123456。注意：由于 hydra 的默认线程不是 1，为了更加便于后面数据包的分析，破解 mssql 时最好将线程数设为 1。

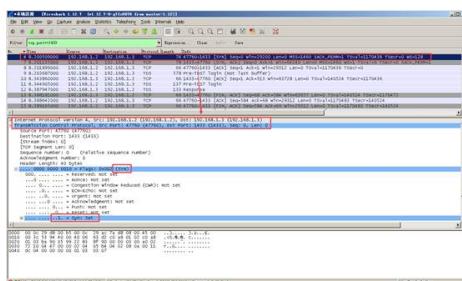
```
root@localhost:~/桌面# hydra -L userlist.txt -P passlist.txt 192.168.1.3 mssql -t 1
Hydra v8.3 (c) 2016 by van Hauser/TMC - Please do not use in military or secret service organizations, or for illegal purposes.
Hydra (http://www.thc.org/thc-hydra) starting at 2017-04-24 12:21:49
[DATA] max 1 task per 1 server, overall 64 tasks, 195 login tries (l:15/p:13), -3 tries per task
[DATA] attacking service mssql on port 1433
[DATA] attacking host: 192.168.1.3  login: sa password: 123456
1 of 1 targets successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2017-04-24 12:22:27
root@localhost:~/桌面#
```

返回 Windows server 2003，wireshark 停止抓包，过滤端口为 1433 的数据包。

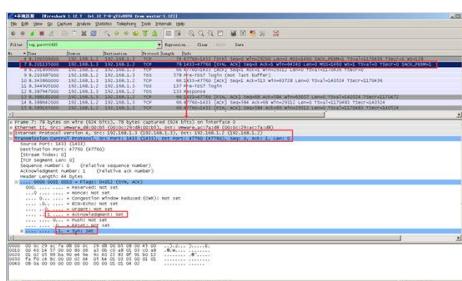


1.4.4.2 分析抓取的数据包，理解 mssql 破解过程

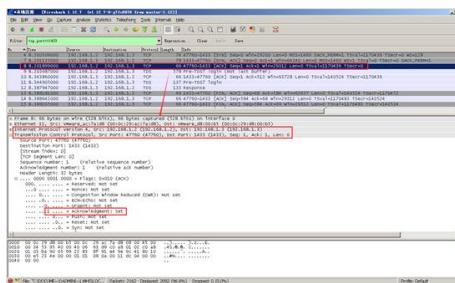
首先查看第一个数据包，源地址为 192.168.1.2，源端口为 47366，目的地址为 192.168.1.3，目的端口为 1433，TCP 序号为 SYN，且置为 1，表示客户端 192.168.1.2 向服务器 192.168.1.3 发送了一个连接请求报文。



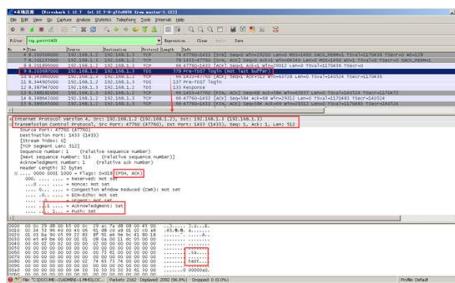
再查看第二个数据包，192.168.1.3 收到请求后确认联机信息，向 192.168.1.2 发送数据包，SYN 置为 1，ACK 也置为 1。



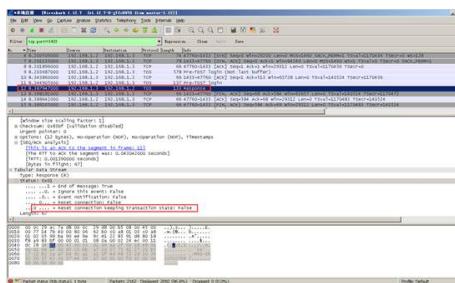
查看第三个数据包，主机 192.168.1.2 收到数据包后检查无误，向 192.168.1.3 发送数据包，ACK 置为 1，至此三次握手完毕，连接建立，后面就可以开始传输数据了。



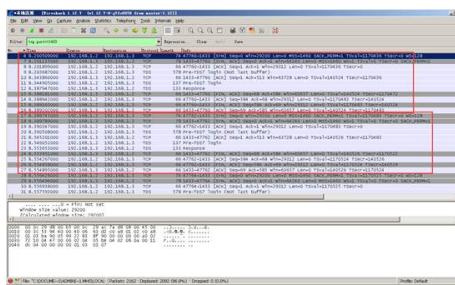
查看第四个数据包，可以看到传输的数据，其中包括了用户名 sa 和密码 test，表明正在破解 mssql。



查看 response，可以看到为 false，说明连接不成功，即用户名或者密码错误。之后则是连接的结束过程。

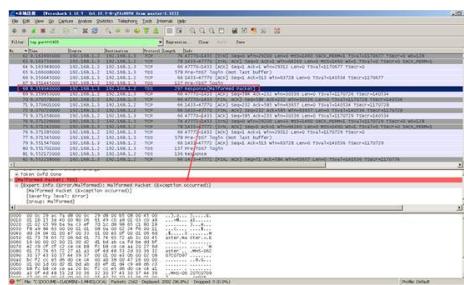
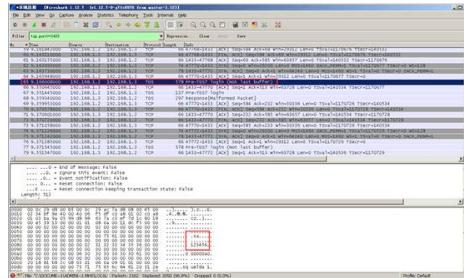


通过分析抓取的数据包，我们可以得知，重复的过程实际上也就是 mssql 破解的过程。



我们找到破解出来的 sa 密码 123456 对应的数据包，并查看其 response 数据包，发现与其

它 response 数据包相比，多了一些数据，说明 sa 的密码很有可能是 123456，实际上也的确如此。



1.4.4.3 查看 mssql 日志

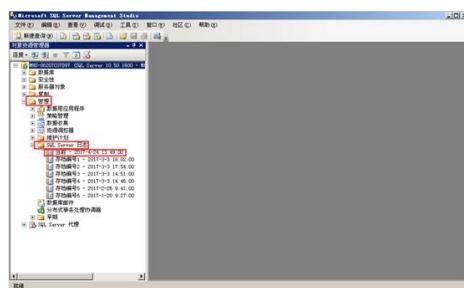
单击“开始”→“所有程序”→“Microsoft SQL Server 2008 R2”→“SQL Server Management Studio”。



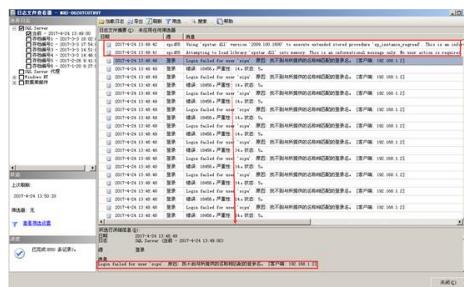
以“Windows 身份认证”方式进行登录（也可用前面破解出来的 sa/123456 进行登录，不过登录方式要选择 SQL Server 身份认证）。



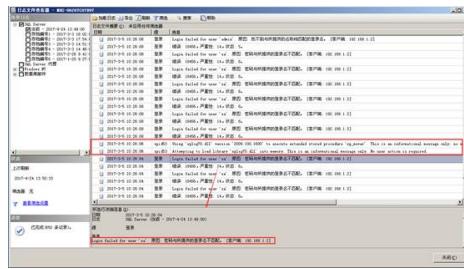
展开“管理”→“SQL Server 日志”，双击当前日志。



在打开的日志文件查看器中查看到的 SQL Server 日志信息中，可以看到前面破解过程中尝试登录的情况，例如以 xipu 用户名进行登录时，提示“找不到与所提供的名称相匹配的登录名”，说明登录名中没有 xipu。其中，前两条日志记录是使用 Windows 身份认证进行登录的信息。看到这么多短时间内登录失败的日志信息，我们就可以得知数据库遭受了攻击，需要采取一定的手段进行防御。



使用 sa 尝试进行登录时，提示“密码与所提供的登录名不匹配”，说明 sa 是一个登录名。在使用 sa 破解的过程中，有两条信息与使用 Windows 身份认证进行登录时的日志信息很相似，我们可以猜测该记录对应的 sa 密码是正确的，实际上也的确如此。



1.4.5 实验目的-了解 MySQL 日志

了解 MySQL 日志.

1.4.6 实验原理

日志是 mysql 数据库的重要组成部分。日志文件中记录着 mysql 数据库运行期间发生的变化，也就是说用来记录 mysql 数据库的客户端连接状况、SQL 语句的执行情况和错误信息等。当数据库遭到意外的损坏时，可以通过日志查看文件出错的原因，并且可以通过日志文件进行数据恢复。

1.4.7 实验环境

CentOS6.5:192.168.1.3 MySQL

1.4.8 实验步骤

1.4.8.1 MySQL 日志介绍

MySQL 有以下几种日志：

- 错误日志: log-err
 - 查询日志: log
 - 慢查询日志: log-slow-queries
 - 二进制日志: log-bin
 - 更新日志: log-update

错误日志：记录 MySQL Server 启动和关闭的详细信息、以及运行过程中较为严重的警告和错误信息，修改错误日志的地址可以在 /etc/my.cnf 中添加

-log-error = [filename]

来设置 mysql 错误日志，在 mysql 中是默认开启错误日志的。

```
root@localhost:~# mysql -u root -p
Enter password: 123456
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 222
Server version: 5.1.73 Source distribution

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show variables like 'log_error';
+-----+-----+
| Variable_name | Value   |
+-----+-----+
| log_error    | /var/log/mysqld.log |
+-----+-----+
1 row in set (0.00 sec)

mysql>
```

```
root@localhost:~# cat /var/log/mysqld.log
[root@localhost ~]# [root@localhost ~]# cat /var/log/mysqld.log
170424 14:32:53 mysqld_safe Starting mysqld daemon with databases from /var/lib/mysql
170424 14:32:53 InnoDB: Initializing buffer pool size = 8.0M
170424 14:32:53 InnoDB: Completed initialization of buffer pool
InnoDB: The first specified data file ./ibdat1 did not exist:
InnoDB: A new database was created based on the default settings.
170424 14:32:53 InnoDB: Setting file ./ibdat1 size to 10 MB
170424 14:32:53 InnoDB: Database physically writes the file full: wait.
170424 14:32:53 InnoDB: Log file ./ib_logfile0 did not exist: new to be created
InnoDB: Setting log file ./ib_logfile0 size to 5 MB
InnoDB: Database physically writes the file full: wait..
170424 14:32:53 InnoDB: log file ./ib_logfile1 did not exist: new to be created
InnoDB: Database physically writes the file full: wait..
InnoDB: Doublewrite buffer found: creating new
InnoDB: Doublewrite buffer created
InnoDB: Creating foreign key constraint system tables
InnoDB: Foreign key constraint system tables created
170424 14:32:53 InnoDB: Creating system tables and triggers
170424 14:32:53 [Note] Event Scheduler: Loaded 0 events
170424 14:32:53 [Note] /usr/libexec/mysqld: ready for connections
Version: '5.1.73' socket: '/var/lib/mysql/mysql.sock' port: 3306  Source distribution
```

查询日志：记录 mysql 的日常日志，包括查询、修改、更新等的每条 SQL 语句信息。使用

```
show global variables like '%genera%';
```

查看 mysql 是否启用了查询日志。可以看到此时的查询日志是关闭的。

```
mysql> show global variables like 'general%';
+-----+-----+
| Variable_name | Value   |
+-----+-----+
| general_log   | OFF    |
| general_log_file | /var/run/mysqld/mysqld.log |
+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

mysql 打开 general log 日志后，所有的查询语句都可以在 general log 文件中输出，如果打开，文件会非常大，建议调试的时候打开，平时关闭。查询日志的输出文件可以在 /etc/my.cnf 中添加

```
general-log-file = [filename]
```

来开启，也可以使用命令

```
set global general_log = on;
set global general_log = off;
```

在 mysql 中打开和关闭。

```

root@localhost:~/桌面
mysql> set global general_log = on;
Query OK, 0 rows affected (0.00 sec)

mysql> show global variables like '%genera%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| general_log | ON   |
| general_log_file | /var/run/mysql/mysqld.log |
+-----+-----+
2 rows in set (0.00 sec)

mysql> set global general_log = off;
Query OK, 0 rows affected (0.00 sec)

mysql> show global variables like '%genera%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| general_log | OFF  |
| general_log_file | /var/run/mysql/mysqld.log |
+-----+-----+
2 rows in set (0.00 sec)

mysql>

```

开启查询日志并新建一个名为 simpleware 的数据库。再关闭查询日志，可以看到在查询日志中记录了创建数据库 simpleware 和关闭查询日志的 SQL 语句。

```

root@localhost:~/桌面
mysql> set global general_log = on;
Query OK, 0 rows affected (0.00 sec)

mysql> create database simpleware;
Query OK, 1 row affected (0.00 sec)

mysql> set global general_log = off;
Query OK, 0 rows affected (0.00 sec)

mysql>

```

```

[root@localhost ~]# cat /var/run/mysql/mysqld.log
/usr/libexec/mysqld, Version: 5.1.73 (Source distribution). started with:
Tcp port: 3306 Unix socket: /var/lib/mysql/mysql.sock
Time: 16:08:52 Id: Command  Argument
170424 16:08:52 222 Query  show global variables like '%genera%'
170424 16:08:58 222 Query  set global general_log = off
/usr/libexec/mysqld, Version: 5.1.73 (Source distribution). started with:
Tcp port: 3306 Unix socket: /var/lib/mysql/mysql.sock
Time: 16:12:24 Id: Command  Argument
170424 16:12:24 222 Query  create database simpleware
170424 16:15:36 222 Query  set global general_log = off
[root@localhost ~]#

```

慢查询日志：慢查询日志中记录的是执行时间较长的 query，可以设一个阀值、将运行时间超过该值的所有 SQL 语句都记录到慢查询日志文件中。该阀值可以通过参数 `long_query_time` 来设置，默认是 10 秒。注意：对于运行时间正好等于 `long_query_time` 的情况并不会被记录，因为在源代码里是判断大于 `long_query_time`，而非大于等于。

可以输入命令

```
show variables like '%slow%';
```

查看慢查询日志功能是否开启，可以看到此时慢查询日志功能并未开启。

```

root@localhost:~/桌面
mysql> show variables like '%slow%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| log_slow_queries | OFF  |
| slow_launch_time | 2   |
| slow_query_log | OFF  |
| slow_query_log_file | /var/run/mysql/mysqld-slow.log |
+-----+-----+
4 rows in set (0.00 sec)

mysql>

```

可使用命令

```
set global slow_query_log = on;
set global slow_query_log = off;
```

开启或关闭慢查询功能。

The screenshot shows a terminal window titled 'root@localhost:~/桌面'. The MySQL prompt 'mysql>' is visible. The user runs two commands: 'set global slow_query_log = on;' and 'set global slow_query_log = off;'. Both commands are highlighted with red boxes. After each command, the output shows the current value of the 'slow_query_log' variable. In the first run, it shows 'ON' and the log file path '/var/run/mysqld/mysqld-slow.log'. In the second run, it shows 'OFF'.

```
mysql> set global slow_query_log = on;
Query OK, 0 rows affected (0.00 sec)

mysql> show variables like 'slow%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| log_slow_queries | ON |
| slow_launch_time | 2 |
| slow_query_log | ON |
| slow_query_log_file | /var/run/mysqld/mysqld-slow.log |
+-----+-----+
4 rows in set (0.00 sec)

mysql> set global slow_query_log = off;
Query OK, 0 rows affected (0.00 sec)

mysql> show variables like 'slow%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| log_slow_queries | OFF |
| slow_launch_time | 2 |
| slow_query_log | OFF |
| slow_query_log_file | /var/run/mysqld/mysqld-slow.log |
+-----+-----+
4 rows in set (0.00 sec)

mysql>
```

二进制日志文件：记录与修改有关的信息，影响数据潜在的内容的信息，二进制日志也叫复制日志，默认在数据目录下，专门查看 mysql 二进制日志文件的命令是 `mysqlbinlog`。

MySQL 记录二进制日志的格式有三种：

1. 基于语句：statement，每一条会修改数据的 sql 都会记录到 master 的 binlog 中，slave 在复制的时候，sql 进程会解析成和原来 master 端相同的 sql 再执行。
2. 基于行的：row，日志中会记录成每一行数据被修改的形式，然后在 slave 端再对相同的数据进行修改，只记录要修改的数据，只有 value，不会有 sql 多表关联的情况。
3. 混合模式：mixed，由 MySQL 自己判断以什么方式记录日志，MySQL 会根据执行的每一条具体的 SQL 语句来区分对待记录的日志形式，也就是在 statement 和 row 之间选择一种。

使用命令

```
show variables like '%log_bin%';
```

查看 mysql 二进制日志文件的配置情况，`log_bin` 用于设定是否启用二进制日志功能，可以看到此时二进制日志功能未开启。

The screenshot shows a terminal window titled 'root@localhost:~/桌面'. The MySQL prompt 'mysql>' is visible. The user runs the command 'show variables like 'log_bin'' to check the configuration of the log_bin variable. The output shows the variable 'log_bin' with a value of 'OFF'.

```
mysql> show variables like 'log_bin';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| log_bin | OFF |
+-----+-----+
4 rows in set (0.00 sec)

mysql>
```

1.4.9 实验目的-MySQL 日志

了解 MySQL 日志。

1.4.10 实验原理

日志是 mysql 数据库的重要组成部分。日志文件中记录着 mysql 数据库运行期间发生的变化，也就是说用来记录 mysql 数据库的客户端连接状况、SQL 语句的执行情况和错误信息等。当数据库遭到意外的损坏时，可以通过日志查看文件出错的原因，并且可以通过日志文件进行数据恢复。

1.4.11 实验环境

Kali Linux: 192.168.1.2 hydra CentOS6.5:192.168.1.3 MySQL

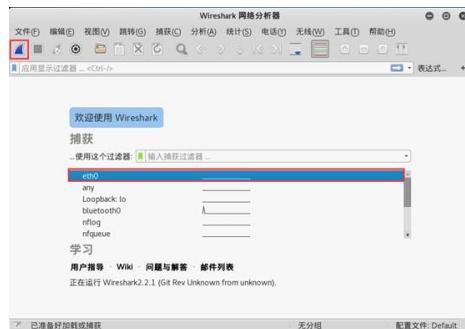
1.4.12 实验步骤

1.4.12.1 破解 mysql 的 root 密码，并抓取破解数据包

在实验开始前先开启 CentOS6.5 上的通用日志。在 kali linux 的终端中输入命令“wireshark”，回车，打开 wireshark。



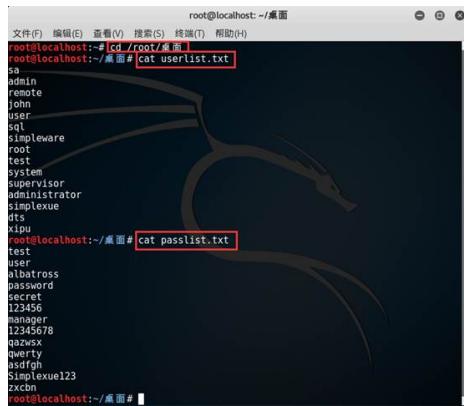
在打开的 wireshark 中选择要监听的网卡，单击“开始捕获分组”按钮，开始进行抓包。



重新打开一个终端，输入命令

```
cd /root/桌面
cat userlist.txt
cat passlist.txt
```

查看用户名文件和密码文件。



在终端输入命令

```
hydra -L userlist.txt -P passlist.txt 192.168.1.3 mysql -t 1
```

开始破解 mysql。这里为了便于后面的数据包分析，将线程数设置为 1。



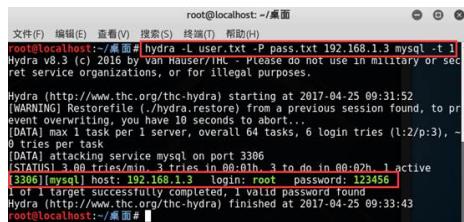
在破解的过程中发现耗时估计会很长，请耐心等待。这里我们为了便于分析，新建并简化用户名文件和密码文件。



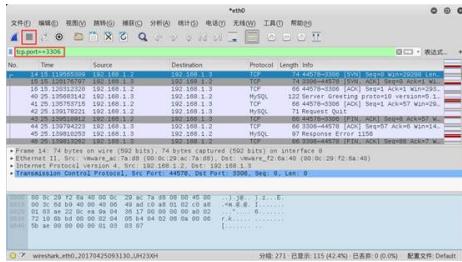
重新开始抓包，在终端输入命令

```
hydra -L user.txt -P pass.txt 192.168.1.3 mysql -t 1
```

开始破解 mysql。

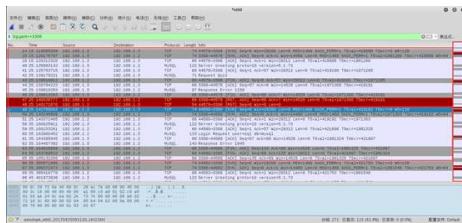


wireshark 停止抓包，过滤端口为 3306 的数据包。

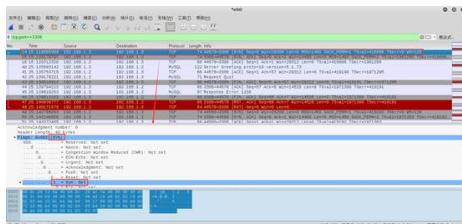


1.4.12.2 分析抓取的数据包，理解 mysql 破解过程

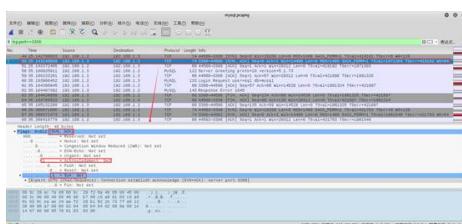
通过观察 wireshark 抓取的数据包可知，wireshark 已经对抓取的数据包使用“[”的符号对连接建立、传输数据、断开连接的过程进行了标识。



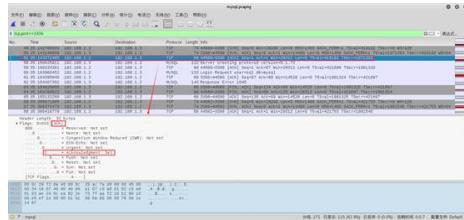
下面我们对其中一个过程的数据包进行分析。首先查看第一个数据包，源地址为 192.168.1.2，源端口为 44578，目的地址为 192.168.1.3，目的端口为 3306，TCP 序号为 SYN，且置为 1，表示客户端 192.168.1.2 向服务器 192.168.1.3 发送了一个连接请求报文。



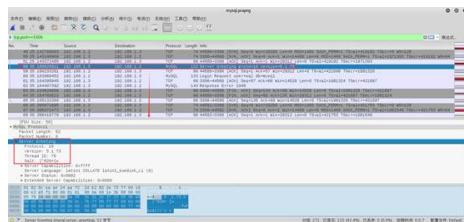
再查看第二个数据包，192.168.1.3 收到请求后确认联机信息，向 192.168.1.2 发送数据包，SYN 置为 1，ACK 也置为 1。



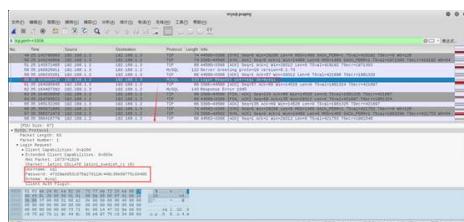
查看第三个数据包，主机 192.168.1.2 收到数据包后检查无误，向 192.168.1.3 发送数据包，ACK 置为 1，至此三次握手完毕，连接建立，后面就可以开始传输数据了。



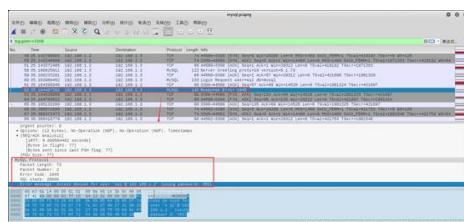
从这个数据包信息可知，mysql 的版本为 5.1.73。



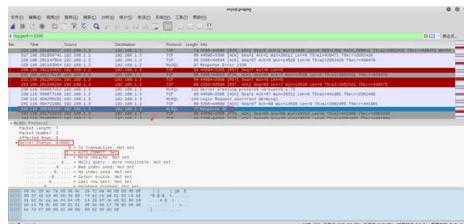
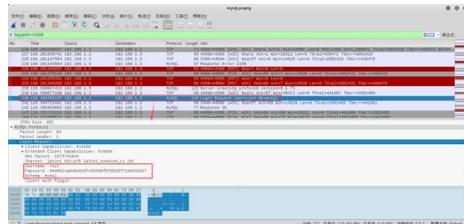
查看这个数据包，可以看到传输的数据，即 mysql 登录信息，用户名为 sql，表明正在破解 mysql。由上一步骤的数据包信息可知 mysql 对密码进行了加盐。



查看 Response，可以看到为 Error，说明连接不成功，即用户名或者密码错误。之后则是连接的结束过程。



我们找到破解出来的 root 密码对应的数据包，并查看其 response 数据包，发现与其它 response 数据包相比，为 OK。



1.4.12.3 查看 mysql 通用日志

在 CentOS6.5 上打开终端，输入命令

```
service mysqld start
mysql -u root -p
# 密码 123456
```



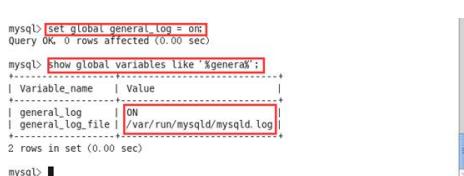
在 mysql 中输入命令

```
set global general_log = on;
```

开启通用日志，再输入命令

```
show global variables like '%genera%';
```

进行查看，可以看到通用日志已开启。



在终端输入命令

```
cat /var/run/mysqld/mysqld.log
```

查看到前面破解过程的 mysql 登录信息。

1.4.13 实验目的-了解 oracle 日志文件

了解 oracle 日志文件.

1.4.14 实验原理

1. 用户对 oracle 数据库的一切操作都记录在数据库的日志文件中，通过日志文件可以查看用户对数据库进行了哪些操作
 2. oracle 分三大类：
 - Alert log files (告警日志)、
 - Trace files (跟踪日志： 用户和进程)
 - redo log (重做日志： 记录数据库的更改)。

1.4.15 实验环境

Windows server 2008 R2:192.168.1.3 oracle 11g

1.4.16 实验步骤

1.4.16.1 告警日志

告警日志文件是一类特殊的跟踪文件(trace file)。告警日志文件命名一般为 `alert_<SID>.log`，其中 SID 为 ORACLE 数据库实例名称。数据库告警日志是按时间顺序记录 message 和错误信息。

在 ORACLE 10g 中，BACKGROUND_DUMP_DEST 参数确定了告警日志的位置，但是告警日志的文件名无法修改。BACKGROUND_DUMP_DEST 参数是动态的。告警日志以及所有后台跟踪文件都会被写至 BACKGROUND_DUMP_DEST 参数所指定的目录。

在 ORACLE 11g 以及 ORACLE 12c 中，告警日志文件的位置有了变化。主要是因为引入了 ADR (Automatic Diagnostic Repository: 一个存放数据库诊断日志、跟踪文件的目录)，关于 ADR 对应的目录位置可以通过查看 `v$diag_info` 系统视图。

在 cmd 中输入命令

```
sqlplus sys/Simplexue123 as sysdba
```

登录 oracle 数据库。

C:\Users\Administrator>sqlplus sys/Simplexue123 as sysdba
SQL*Plus: Release 11.2.0.1.0 Production on 星期三 4月 26 10:09:55 2017
Copyright (c) 1982, 2010, Oracle. All rights reserved.

连接到:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
SQL>

在 SQL 中输入命令

```
select * from v$diag_info;
```

查看 ADR 对应的目录位置，可以看到 Diag Trace 对应的目录为文本格式的告警日志文件所在的目录，而 Diag Alert 对应的目录为 XML 格式的警告日志（对应为 log.xml）。

INST_ID	NAME	VALUE
TRUE	1 Diag Enabled	
	1 ADR Base	c:\app\administrator
	1 ADR Home	c:\app\administrator\diag\rdbms\orcl\orcl
	INST_ID	NAME
	VALUE	
	1 Diag Trace	c:\app\administrator\diag\rdbms\orcl\orcl\trace
	1 Diag Alert	c:\app\administrator\diag\rdbms\orcl\orcl\alert
	1 Diag Incident	c:\app\administrator\diag\rdbms\orcl\orcl\incident

INST_ID	NAME	VALUE
	1 Diag Cdump	c:\app\administrator\diag\rdbms\orcl\orcl\cdump
	1 Health Monitor	c:\app\administrator\diag\rdbms\orcl\orcl\hm
	1 Default Trace File	c:\app\administrator\diag\rdbms\orcl\trace\orcl_ora_1212.trc
	INST_ID	NAME
	VALUE	
0	1 Active Problem Count	0
0	1 Active Incident Count	0

已选择11行。
SQL>



告警日志包含了下面一些内容的信息。像一些 ORA 错误，对于监控数据库有极其重要的作用。

1. 所有的内部错误(ORA-600)信息, 块损坏错误(ORA-1578)信息, 以及死锁错误(ORA-60)信息等。
2. 管理操作, 例 CREATE、ALTER、DROP 语句等, 以及数据库启动、关闭以及日志归档的一些信息。包括以下两类:
 - 涉及物理结构的所有操作: 例如创建、删除、重命名数据文件与联机重做日志文件的 ALTER DATABASE 命令, 此外还涉及重新分配数据文件大小以及将数据文件联机与脱机的操作。
 - 表空间操作: 例如 DROP 与 CREATE 命令, 此外还包括为了进行用户管理的备份而将表空间置入和取出热备份模式的操作
3. 与共享服务器或调度进程相关功能的消息和错误信息。
4. 物化视图的自动刷新过程中出现的错误。
5. 动态参数的修改信息。

既然告警日志如此重要, 而我们也不可能随时手工去查看告警日志文件, 那么我们就必须监控告警日志, 那么监控告警日志有哪些方案呢? 有以下 3 种方案:

- 在 ORACLE 10g, 可以将告警日志文件信息读入全局临时表, 然后就可以定制一些 SQL 语句查询告警日志的信息。但这个方案有几个不足之处, 一是如果数据库宕机了的情况下, 是无法获取这些错误信息的, 因此有些特定场景不适用。二是日志文件比较大的时候, 监控告警日志信息比较频繁的时候, 会产生不必要的 IO 操作。
- 通过外部表来查看告警日志文件的内容, 相当的方便, 与方案一一样也是使用定制 SQL 语句来查询错误信息。
- 将告警日志进行归档。告警日志文件如果不加管理的话, 那么文件会持续增长, 有时候文件会变得非常大, 不利于读写。一般建议将告警日志按天归档, 归档文件保留三个月(视情况而定), 可通过编写脚本来进行归档。

1.4.16.2 跟踪日志

跟踪文件（trace file）的作用，通常是一个服务器进程对某种异常错误条件做出响应时创建的诊断文件。一个 sql 发到数据库，肯定会有个“接收请求 → 数据处理 → 返回应答”的过程。那么就需要不同的进程来执行相应的步骤。如果进程在执行过程中发生错误，那么就会记录到跟踪文件中。

在创建数据库的过程中遇到的错误，可以通过查找 Oracle 数据库的告警日志文件获得，在某些情况下，还会有详细的跟踪文件生成，这些文件的位置，在 Oracle 11g 之前，由 *dump 参数指定，告警日志文件 alert_<ORACLE_SID>.log 的位置由参数 background_dump_dest 定义。

跟踪文件的查看命令为

```
show parameter SQL_TRACE;
```

如果结果为 FALSE，可使用命令

```
alter system set SQL_TRACE = TRUE SCOPE = both;
```

来设置打开，使用命令

```
alter system set SQL_TRACE = FALSE;
```

来进行关闭。

NAME	TYPE	VALUE
sql_trace	boolean	FALSE

NAME	TYPE	VALUE
sql_trace	boolean	TRUE

NAME	TYPE	VALUE
sql_trace	boolean	FALSE

在 SQL 中输入命令

```
show parameter DUMP_DEST;
```

查看跟踪文件位置，会看到有三个跟踪文件目录。

NAME	TYPE	VALUE
background_dump_dest	string	c:\app\administrator\diag\rdbms\sql1\trace
core_dump_dest	string	c:\app\administrator\diag\rdbms\sql1\dump
user_dump_dest	string	c:\app\administrator\diag\rdbms\sql1\trace

其中，background_dump_dest 为后台转储，core_dump_dest 为内核转储，user_dump_dest 为用户转储，一般而言，我们只对后台和用户转储目标感兴趣。

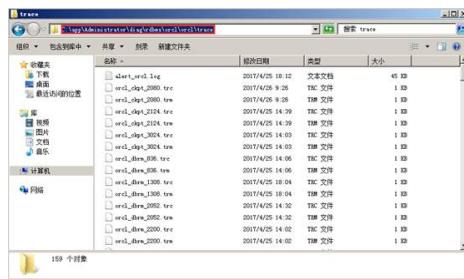
在 SQL 中输入命令

```
show parameter background_dump_dest;
```

查看后台转储跟踪日志文件的路径。

```
SQL> show parameter background_dump_dest;
NAME          TYPE        VALUE
background_dump_dest    string      c:\app\administrator\diag\rdbms
                           $orcl$orcl\trace
SQL>
```

可知跟踪日志文件所在的路径,可以在该路径下,找到日志文件。跟踪文件名形如:`orcl_cjq0_2348.trc`(`SID 名 + 进程名 + 进程 ID`)。



1.4.16.3 重做日志

重做日志分为在线重做日志和归档重做日志。重做日志的简单原理：在数据更新操作 commit 前，将更改的 SQL 脚本写入重做日志。主要用于数据库的增量备份和增量恢复。重做日志直接对应于硬盘的重做日志文件（有在线和归档两种），重做日志文件以组（Group）的形式组织，一个重做日志组包含一个或者多个日志文件。

online Redo log files: 在线重做日志，又称联机重做日志，指 Oracle 以 SQL 脚本的形式实时记录数据库的数据更新，换句话说，实时保存已执行的 SQL 脚本到在线日志文件中（按特定的格式）。

对于在线重做日志，Oracle 11g 默认对于每个数据库实例，建立 3 个在线日志组，每组一个日志文件，文件名称为 REDO01.LOG, REDO02.LOG 和 REDO03.LOG。(用户可以通过视图操作添加/修改/删除日志组和日志文件来自定义在线重做日志)

每组内的日志文件的内容完全相同，且保存在不同的位置，用于磁盘日志镜像，以做多次备份提高安全性。默认情况这 3 组通常只有一组处于活动状态，不断地同步写入已操作的脚本，当日志文件写满时（达到指定的空间配额），如果当前数据库处于归档模式，则将在线日志归档到硬盘，成为归档日志；若当前数据库处于非归档模式，则不进行归档操作，而当前在线日志的内容会被下一次重新写入覆盖而无法保存。因此，通常数据库在运行时，是处于归档模式下的，以保存数据更新的日志。

当前归档日志组写满后，Oracle 会切换到下一日志组，继续写入，就这样循环切换；当处于归档模式下，切换至原已写满的日志组，若该日志组归档完毕则覆盖写入，若没有则只能使用日志缓冲区，等待归档完毕之后才能覆盖写入。当然，处于非归档模式下是直接覆盖写入的。

Oracle 提供了 2 个视图用于维护在线重做日志：V\$LOG 和 V\$LOGFILE，我们可以通过这两个视图查看和修改在线日志。

在 SQL 中输入命令

```
SELECT * FROM v$log;
```

通过 v\$log 视图查询在线日志的总体信息。

SQL> SELECT * FROM v\$log;									
GROUP#	THREAD#	SEQUENCE#	BYTES	BLOCKSIZE	MEMBERS	ARC	STATUS	FIRST_CHANGE#	FIRST_TIME
CURRENT	1	7	52428800	512	1	NO	1	2,8147E+14	25-4月 -17
INACTIVE	1	1080112	52428800	512	1	NO	2	1046702	25-4月 -17
INACTIVE	1	1015797	52428800	512	1	NO	3	1046702	25-4月 -17
INACTIVE	1	1046702	52428800	512	1	NO	4	1080112	25-4月 -17

为了便于查看，复制到 txt 文件整理后的在线日志的总体信息如下。

GROUP#	THREAD#	SEQUENCE#	BYTES	BLOCKSIZE	MEMBERS	ARC	STATUS	FIRST_CHANGE#	FIRST_TIME	NEXT_CHANGE#	NEXT_TIME
1	1	7	52428800	512	1	NO	CURRENT	1080112	25-4月 -17	2,8147E+14	
2	1	5	52428800	512	1	NO	INACTIVE	1015797	25-4月 -17	1046702	25-4月 -17
3	1	6	52428800	512	1	NO	INACTIVE	1046702	25-4月 -17	1080112	25-4月 -17

在 SQL 中输入命令

```
SELECT * FROM v$logfile ORDER BY group#;
```

通过 v\$logfile 视图查询在线日志文件信息。

SQL> SELECT * FROM v\$logfile ORDER BY group#;									
GROUP#	STATUS	TYPE	MEMBER	IS_	---	1	ONLINE	C:\APP\ADMINISTRATOR\ORADATA\ORCL\REDO01.LOG	NO
1	ONLINE	LOG	1	NO		2	ONLINE	C:\APP\ADMINISTRATOR\ORADATA\ORCL\REDO02.LOG	NO
2	ONLINE	LOG	2	NO		3	ONLINE	C:\APP\ADMINISTRATOR\ORADATA\ORCL\REDO03.LOG	NO
3	ONLINE	LOG	3	NO					

为了便于查看，复制到 txt 文件整理后的在线日志文件信息如下。

GROUP#	STATUS	TYPE	MEMBER	IS_
1	ONLINE	LOG	C:\APP\ADMINISTRATOR\ORADATA\ORCL\REDO01.LOG	NO
2	ONLINE	LOG	C:\APP\ADMINISTRATOR\ORADATA\ORCL\REDO02.LOG	NO
3	ONLINE	LOG	C:\APP\ADMINISTRATOR\ORADATA\ORCL\REDO03.LOG	NO

此外可以通过 ALTER、DATABASE、ADD、DELETE 等命令增加、修改或删除在线日志或日志组。

Archive Redo log files: 归档重做日志，简称归档日志，指当条件满足时，Oracle 将在线重做日志以文件形式保存到硬盘（持久化）。

所谓的归档，就是指将在线日志进行归档、持久化到成固定的文件到硬盘，便于以后的恢复和查询。当然，前提条件是数据库要处于归档模式。

Oracle 11g 默认是为归档日志设定 2 个归档位置，这 2 个归档位置的的归档日志的内容完全一致，但文件名不同。

1.4.17 实验目的-分析 oracle 日志

分析 oracle 日志.

1.4.18 实验原理

1.4.18.1 Oracle 加密原理

当 Oracle 发起连接后，Oracle 客户端向 oracle 数据库发送自己版本号、包含的加密算法等信息。最终服务端确定使用什么加密算法，然后进行 O3logon 验证。O3logon 验证是一种查询-响应协议，它利用 DES 加密技术保护这个会话的密钥 (sesskey)，保证 sesskey 不会在网络中传输，所以即使有人监听网络也不会暴露核心密钥。其中 O3logon 验证的核心是 sesskey。

Oracle 11g 在 10g 的基础上进行了一定的改变。假设我们已经取得一个含有 Oracle 登录信息的网络通讯包。省略掉前面和密码关系不大的信息在数据包中寻找到 4 个相关信息，分别是数据库发送给客户端的 S_AUTH_SESSKEY、AUTH_VFR_DATA、客户端发送给服务器的 C_AUTH_SESSKEY 和 AUTH_PASSWORD。

假设取得了 Oracle_hash，11g 基本同于 10g，客户端和数据库分别以 Oracle_hash 为基础生成 S_AUTH_SESSKEY 和 C_AUTH_SESSKEY。客户端对传过来的 S_AUTH_SESSKEY 做 AES192 解密处理拿到 server_sesskey，把 server_sesskey 和自己的 client_sesskey 做 md5 生成 combine，用 combine 生成 AUTH_PASSWORD。服务器最后用 combine 对 AUTH_PASSWORD 解密，对比密码，如果密码一致则登陆成功。

11g 最大的变化在生成 Oracle_hash 上采取了和 10g 不同的策略。Oracle 11g 为了提高 Oracle_hash 的安全性，引入了 AUTH_VFR_DATA 这个随机值，取消了明文密码。每个会话的 AUTH_VFR_DATA 都不同。从根本上避免 9i、10g 同字符串（用户名 + 密码组成的字符串）带来的无论哪台机器 oracle_hash 一致的巨大安全隐患。从 11g 开始，oracle 和密码相关登陆信息全部采用了密文。有效地加大了破解难度。对于 oracle 安全性问题，一定注意防止网络监听，设计 SID 的时候尽量避免 ORACLE、TEST 等常用名。端口号尽量不要选用 1521 和 1523 来增加扫描难度。使用复杂密码，定期更换密码等都会有助于 oracle 的安全

1.4.18.2 Orabrute

这里我们使用 Orabrute 工具来进行远程破解 oracle，在使用这个工具的时候，需要系统提前安装好 sqlplus，该工具的原理很简单，就是不停的调用 sqlplus 然后进行登录验证，帐户选择的是 sys，密码则为 password.txt 中的密码单词。只要登录成功，就会调用 selectpassword.sql 脚本抓取出在 SYS.USER\$ 表中的其他用户的哈希值，然后退出程序。这里有个值得注意的地

方，当第二次运行 Orabrute 的时候，需要删除或移动同目录下的前一次运行 Orabrute 时生成的 thepasswordsarehere.txt 文件。

1.4.18.3 SQLPLUS

Oracle 的 sqlplus 是与 oracle 数据库进行交互的客户端工具，借助 sqlplus 可以查看、修改数据库记录。在 sqlplus 中，可以运行 sqlplus 命令与 sql 语句。

1.4.19 实验环境

- Windows server 2003: 192.168.1.2 wireshark、orabrute、sqlplus
- Windows server 2008 R2: 192.168.1.3oracle 11g

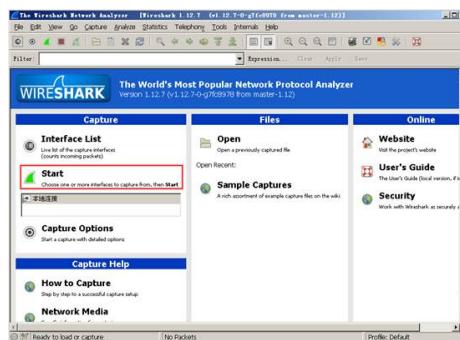
1.4.20 实验步骤

1.4.20.1 破解 oracle 用户 sys 的密码

为了更直观地查看日志，在 Windows server 2008 R2 上，单击“开始”→“管理工具”→“事件查看器”，展开“Windows 日志”，右键“应用程序”，选择清除日志。



在 Windows server 2003 (192.168.1.2) 上打开 wireshark，选择要监听的网卡后，单击“Start”按钮，开始进行抓包。



事先先删除桌面上 tools/oracle 目录下的 thepasswordsarehere.txt 文件，没有则跳过。

打开 cmd，输入命令

```
cd 'C:\Documents and Settings\Administrator\桌面\tools\oracle'
```

再输入命令

```
orabrute 192.168.1.3 1521 orcl 10 | more
```

在破解过程中，可按空格键查看之后的内容。

```
C:\命令提示符
Microsoft Windows (版本 5.2.3790)
(C) 版权所有 1985-2003 Microsoft Corp.

C:\Documents and Settings\Administrator\桌面\tools\oracle>cd C:\Documents and Settings\Administrator\桌面\tools\oracle
C:\Documents and Settings\Administrator\桌面\tools\oracle>orabrute 192.168.1.3 1521 orcl 10 | more
ERROR:
ORA-01017: invalid username/password; logon denied

SP2-0751: Unable to connect to Oracle. Exiting SQL*Plus
ERROR:
ORA-01017: invalid username/password; logon denied

SP2-0751: Unable to connect to Oracle. Exiting SQL*Plus
ERROR:
ORA-01017: invalid username/password; logon denied

SP2-0751: Unable to connect to Oracle. Exiting SQL*Plus
ERROR:
ORA-01017: invalid username/password; logon denied

SP2-0751: Unable to connect to Oracle. Exiting SQL*Plus
Orabrate v 1.2 by Paul M. Wright and David J. Morgan:
orabrate <hostip> <port> <id> <n>(lllitineexit>sqlplus.exe -S -L "SYS/test@192.168.1.3:1521/orcl" as sysdba @selectpassword.sql
sqlplus.exe -S "SYS$USER@192.168.1.3:1521/orcl" as sysdba @selectpassword.sql
-- None --
```

除了在 cmd 中可以查看到破解结果，还可以在 thepasswordsare.txt 文件中进行查看，可以看到 sys 破解出来的密码。

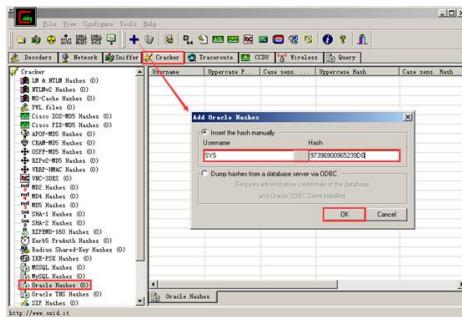
```
C:\命令提示符
SP2-0751: Unable to connect to Oracle. Exiting SQL*Plus
NAME          PASSWORD
SYS           97396900965239D0
PUBLIC
CONNECT
RESOURCE
DBA
SYSTEM        7F32DBB34C75D309
SELECT_CATALOG_ROLE
EXECUTE_CATALOG_ROLE
DELETE_CATALOG_ROLE
OUTLN         4A3B55E0B5959C81
IMP_FULL_DATABASE
NAME          PASSWORD
IMP_FULL_DATABASE
LOGSTBY_ADMINISTRATOR
DBFS_ROLE
DIP          CE4A36B8EB6C859C
AQ_ADMINISTRATOR_ROLE
AQ_USER_ROLE
DATAPUMP_EXP_FULL_DATABASE
DATAPUMP_IMP_FULL_DATABASE
ADM_PARALLEL_EXECUTE_TASK
GATHER_SYSTEM_STATISTICS
JRN_DEPLOY
NAME          PASSWORD
ORACLE_OCM    5A2EB26A915795BC
RECOVERY_CATALOG_OWNER
```

```

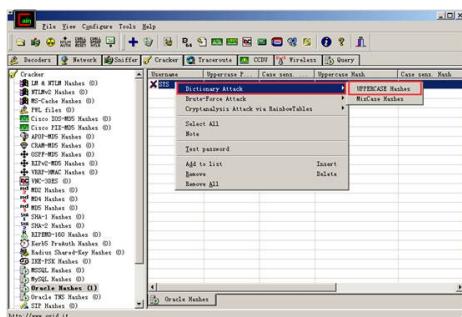
thekeypasswords.txt - 记事本
NAME          PASSWORD
SYS          97396900965239D0
PUBLIC
CONNECT
RESOURCE
DBA
SYSTEM        7F3200B94C7503B9
SELECT_CATALOG_ROLE
EXECUTE_CATALOG_ROLE
DELETE_CATALOG_ROLE
DMLN          A2B8A5E0B595C81
EXP_FULL_DATABASE
NONE          PASSWORD
IMP_FULL_DATABASE
LOGSTBY_ADMINISTRATOR
DBFS_ROLE
DIP          CEA4A36B8E06CA59C
AQ_ADMINISTRATOR_ROLE
AQ_USER_ROLE
DATAPUMP_EXP_FULL_DATABASE
DATAPUMP_IMP_FULL_DATABASE
AQH_PUBLISHING_TASK
GATHER_SYSTEM_STATISTICS
JAVAE_DEPLOY
NONE          PASSWORD

```

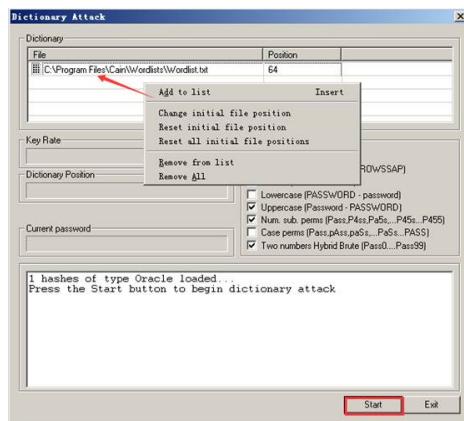
打开 Cain，在 Cracker 下选择 Oracle Hashs，先在空白处单击一下鼠标，再单击“+”添加信息，用户名为 sys，hash 值为 97396900965239D0，单击“OK”。



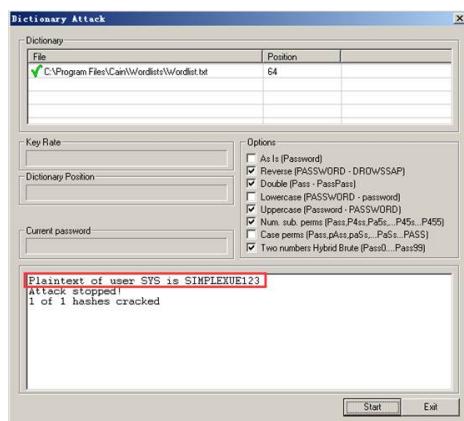
右键添加的条目，选择“Dictionary Attack”→“UPPRECASE Hashs”。



在 Dictionary 下方的空白处右键，选择“Add to list”，添加字典文件后，单击“Start”按钮，开始破解。

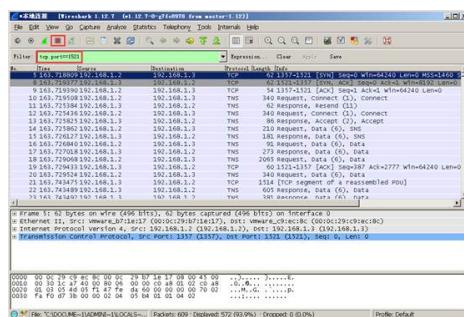


破解出 sys 的密码为 Simplexue123 (这里显示的时候不区分大小写)。

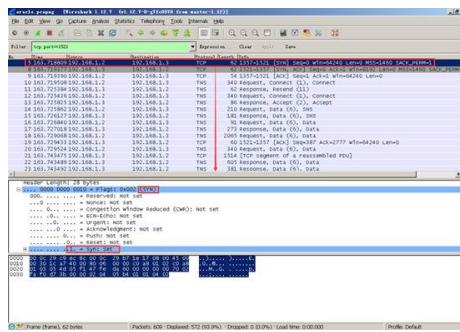


1.4.20.2 理解 oracle 数据库的破解过程

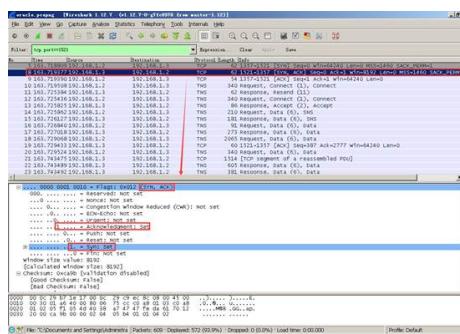
Wireshark 停止抓包，过滤端口号为 1521 的数据包。



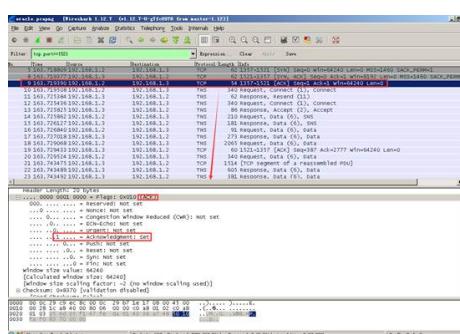
下面我们对其中一个过程的数据包进行分析。首先查看第一个数据包，源地址为 192.168.1.2，源端口为 1357，目的地址为 192.168.1.3，目的端口为 1521，TCP 序号为 SYN，且置为 1，表示客户端 192.168.1.2 向服务器 192.168.1.3 发送了一个连接请求报文。



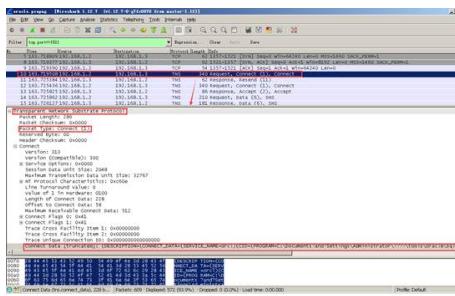
再查看第二个数据包，192.168.1.3 收到请求后确认联机信息，向 192.168.1.2 发送数据包，SYN 置为 1，ACK 也置为 1。



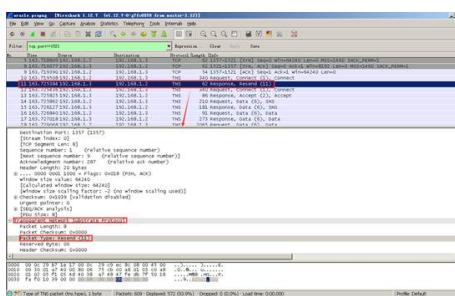
查看第三个数据包，主机 192.168.1.2 收到数据包后检查无误，向 192.168.1.3 发送数据包，ACK 置为 1，至此三次握手完毕，连接建立。



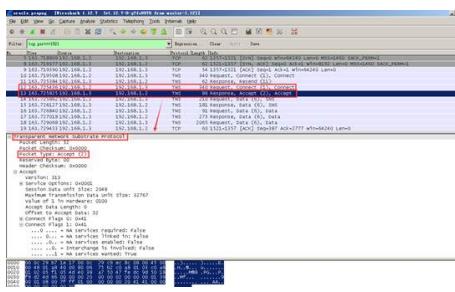
在 TNS (Transparent Network Substrate Protocol) 协议中，每个 TNS 完整数据都包含一个通用包头，说明接受数据的长度及其相关校验和解析的信息，类型 type 的值为 1 时，即 Connect(1)，表示请求连接。



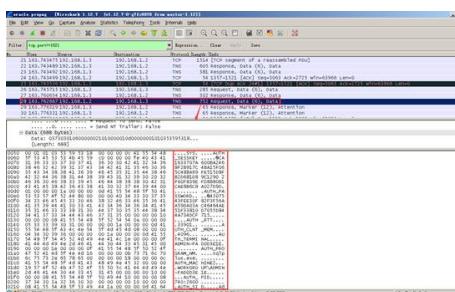
当 type 的值为 11，即 Resend(11) 时，表示重新发送，即服务器要求客户端再发送一次连接请求。



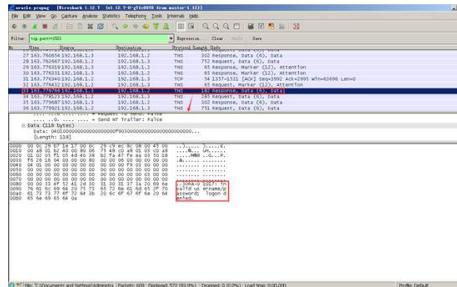
客户端再次发送连接请求。当 type 的值为 2，即 Accept(2) 时，表示接受。



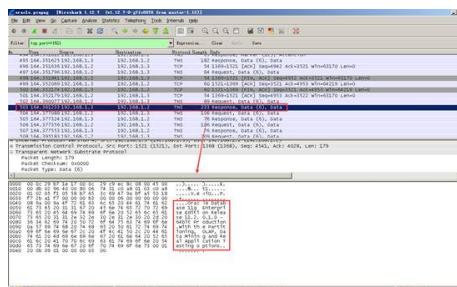
当 type 的值为 6，即 Data(6) 时，表示数据，即进行着数据的传输。继续往下分析数据包，我们找到相关的登录请求信息数据包，我们发现无法看到明文密码，因为 oracle 的登录认证过程密码是加密的，无法查看。



登录失败信息如下。



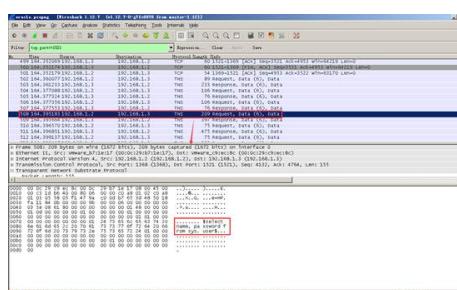
在抓取的破解数据包中，有这样一个数据包，通过分析，可以发现这是登录成功时显示的信息，说明破解出了 sys 的密码，登录成功了。



登录成功后调用 selectpassword.sql 脚本，执行 SQL 语句

```
select name,password from sys.user$;
```

抓取出在 SYS.USER\$ 表中的其他用户的哈希值，并将结果保存在 thepasswordsare.txt 文件中，最后退出程序。



selectpassword.sql 脚本内容如下。

```
select password from sys.user$;
spool thepasswordsare.txt
exit
```

所以，只要在破解过程中，有一个密码可以登录成功，我们可以获得所有的用户名及其密码 hash 值，再使用 Cain 来破解密码。thePasswordsAre.txt 的内容如下。

l:\theusername&password.txt - 记事本
文件(F) 编辑(E) 格式(O) 帮助(H)

NAME	PASSWORD
SYS	9729698996529098
PUBLIC	
CONNECT	
RESOURCE	
DBA	
SYSTEM	7F32D0B9aC750309
SELECT_CATALOG_ROLE	
CREATE_CATALOG_ROLE	
DELETE_CATALOG_ROLE	
OUTLN	4B3B055E0B595C81
EXP_FULL_DATABASE	

NAME	PASSWORD
IMP_FULL_DATABASE	
RECOVERY_ADMINISTRATOR	
DBMS_ROLE	
DIP	C6A936B8E06C059C
ADM_APPADMIN_ROLE	
USER_ROLE	
DATAPUMP_EXP_FULL_DATABASE	
DATAPUMP_IMP_FULL_DATABASE	
ADM_PARALLEL_EXECUTE_TASK	
GATHER_SYSTEM_STATISTICS	
JDBC_DEPLOY	

NAME	PASSWORD

1.4.20.3 查看日志

切换到 Windows server 2008 R2 上，打开 cmd，使用命令

```
sqlplus sys/Simplexue123 as sysdba
```

进入 oracl。

在 oracl 中，使用命令

```
show parameter audit;
```

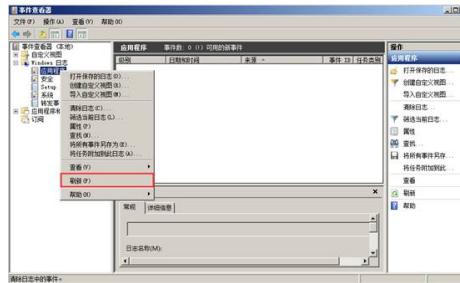
查看和审计相关的主要参数，audit_sys_operations 默认为 false，当设置为 true 时，所有 sys 用户（包括以 sysdba,sysoper 身份登录的用户）的操作都会被记录，audit trail 不会写在 aud\$ 表中，如果是 windows 平台，audit trail 会记录在 windows 的事件管理中，如果是 linux/unix 平台则会记录在 audit_file_dest 参数指定的文件中。所以本实验审计 sys 的信息时使用事件查看器进行查看。

```
管理员: 命令提示符 - sqlplus sys/Simplex123 as sysdba
Microsoft Windows [版本 6.1.7601]
版权所有 © 2009 Microsoft Corporation. 保留所有权利。
C:\Users\Administrator[sqlplus sys/Simplex123 as sysdba]
SQL*Plus: Release 11.2.0.1.0 Production on 星期一 10月 23 07:14:37 2017
Copyright © 1982, 2008, Oracle. All rights reserved.

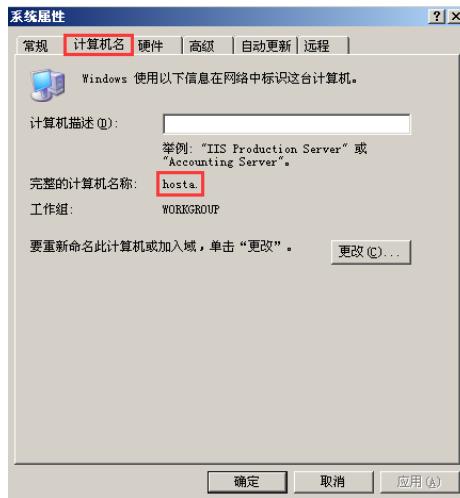
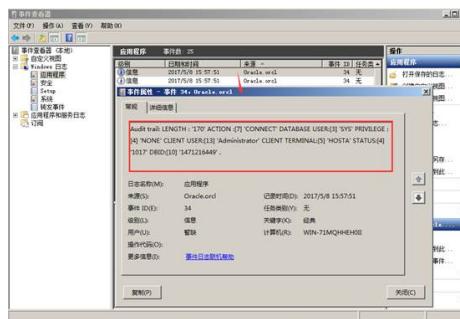
连接到:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
    用 Partitioning, OLAP, Data Mining 和 Real Application Testing options

SQL> show parameter audit;
NAME                           TYPE        VALUE
audit_file_dest                string      C:\APP\ADMINISTRATOR\ADMIN\NORC
audit_sys_operations            boolean    FALSE
audit_trail                     string      DB
SQL>
```

在 Windows server 2008 R2 上刷新“应用程序”日志。



发现有多条新增多条应用程序日志信息，查看某条信息记录。



在查看这些消息记录时，发现有一条记录与其他的记录相比，略有不同，STATUS:[1] '0' 表示数据库连接成功。在短时间内有大量的 oracle 数据库连接审计信息，在大多数连接失败的信息中，有连接成功的记录，我们有理由相信数据库遭受到了暴力破解攻击。

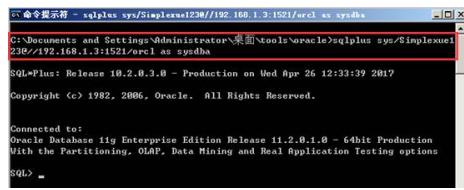
这时我们就需要采取应对的应对措施。例如关闭数据库、更改数据库连接密码、限制登录连接次数等等。



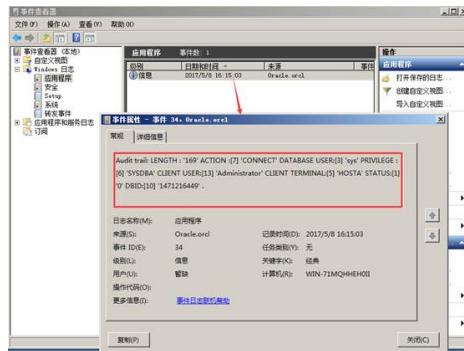
再次清空“应用程序”日志。在 Windows server 2003 (192.168.1.2) 上，在 cmd 中输入命令

```
sqlplus sys/Simplexue123@192.168.1.3:1521/orcl as sysdba
```

使用破解出来的密码进行登录，登录成功。



在 Windows server 2008 R2(192.168.1.3) 上查看“应用程序”日志，发现其信息与我们之前猜测的一样，为登录成功的审计日志信息。



Oracle 11g 中为了防止暴力破解数据库中用户的密码，提供了一种常见手段：延长失败尝试响应。

这种手段的策略是：在连续使用错误密码反复尝试登录时，从第四次错误尝试开始，每次增加 1 秒的延迟，最长延迟目前是 10 秒。

使用这种手段可以相对比较有效的防治用户密码的暴力破解，能够使用到这种手段的前提是 FAILED_LOGIN_ATTEMPTS 参数设置的足够大或无限大，否则用户密码超过 10 次的错误尝试之后该用户将被锁定。

所以本实验为了演示效果较佳，使用的密码文件的密码数较少，便于破解。