

上海交通大学

硕士学位论文

UltraSurf软件的逆向分析技术研究

姓名：张磊

申请学位级别：硕士

专业：密码学

指导教师：谷大武

20090101

## UltraSurf 软件的逆向分析技术研究

### 摘 要

UltraSurf 软件是在互联网客户端运行的应用程序。它使用远程代理服务器和自定义的加密协议，可以突破传统的网络过滤，实现对远程信息的透明访问。本文通过软件逆向工程中的白盒，黑盒分析方式，主要运用 Ollydbg 动态反汇编调试器、Ethereal 网络抓包软件以及 Iptables 防火墙等工具，从工作过程、加解密方式和算法、网络连接手段等方面对 UltraSurf 软件进行了分析。得出了该软件的工作原理、与代理服务器的加密通信方法、动态获取加密代理服务器地址的方式等。

根据对软件分析结果，我们设计了一套封堵方案，并将其部署在实验室的网络中进行了验证。实验结果表明，该方案能够有效地阻止这类软件获得加密代理地址，从而使其无法使用，同时用户的正常网络通信不受影响。

在对 UltraSurf 分析的基础上还总结出了穿透类软件的特点及一般分析方法。

**关键词：**网络监控，反汇编，加密代理

# **Research and Analysis of UltraSurf Software by Reverse Engineering**

## **ABSTRACT**

UltraSurf is a well-know client application on the Internet. With the help of its private communication protocols and remoting servers as agents, it can be used to penetrate through the network control available, so as to make it accessible to remote information. This thesis analyzes the UltraSurf (version 8.8) by using tools, such as Ollydbg, Ethereal and Iptables. The main method includes “White Box” and “Black Box” of the software reverse engineering. The analysis concentrated on the working process, methods and algorithms of encryption and decryption, Internet connection of the software, and the analysis result includes the working principle of the software, the way to encrypt the communication between the machine and the proxy servers, and dynamic methods to get the IP address of the proxy servers.

From the analysis result, a scheme to control the behavior of the UltraSurf was set up. We validate it by deploying the system in the lab network environment. The rest result of the current control system indicates that the current control system could make the users in the test environment unable to use UltraSurf, but browse other websites as usual.

We also summarize the characteristics of this kind of software and raise a general analytical method based on the analysis of UltraSurf.

**KEY WORDS:** Network Monitoring, Disassembly, Secure Proxy

上海交通大学硕士学位论文

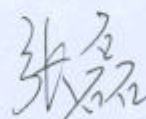
UltraSurf 软件的逆向分析技术研究

上海交通大学

## 学位论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：



日期：2019年1月12日

## 第一章 绪论

### 1.1 研究背景及意义

21 世纪已经成为真正意义上的信息时代，互联网在其中发挥了不可估量的信息传播作用，极大地促进了社会的进步和发展。但是通过互联网传递的信息鱼龙混杂，需要有方法加以鉴别，否则将会对社会造成危害。目前，互联网上传播不良信息的各类网站还很多，各国互联网管理部门都采取了监视、过滤、封锁等一系列手段，以此来净化互联网环境，抵御不良信息。

近年来，我国出现了一类可以突破现有封锁手段连接到发布不良信息网站的软件，运行这类软件即可使用 IE 浏览器连接到这些网站浏览信息。这些软件的典型代表有无界浏览 (UltraSurf)、自由门 (FreeGate)、花园 (Garden) 等。

这类软件虽然名称不同，但是内部的原理有很多相似之处，所以将它们统称为“穿透类软件”。

对这些软件给予关注和分析有利于对它们的原理进行深入了解，从而达到控制软件行为、净化网络信息、保障我国信息安全的目的。

### 1.2 研究现状

从 90 年代开始，我国有关部门对不良主机、代理服务器、以及某些 DNS 服务器进行封锁的有效手段是过滤其 IP 地址和端口，但这种技术不能确定哪些站点是不良站点，而且对那些经常“跳变的”不良站点的过滤无能为力。

2002 年以后，关键字/词过滤逐渐成为我国互联网信息内容控制的主流技术，该技术通过采用人工智能和自动识别等方法，对网址和网页中的内容进行过滤。关键字/词过滤的弱点是对加密信息无能为力，而不公开的加密手段是很多突破网络封锁软件的基础。域名劫持是对互联网不良信息进行封锁的另一种常用手段，即利用自己网络范围内的域名解析服务器，把被访问的域名解析成假的 IP 地址，让用户无法正确访问所请求的资源。域名劫持的主要弱点是它不很稳定，另外对加密后的域名也毫无办法。

近年来出现的穿透类软件，如 UltraSurf 等，采用了加密代理技术，其原理

是：通过主机安装的特定软件将访问请求的信息加密后传给远端的加密代理服务器，服务器解密并完成该访问请求，然后将访问内容加密后传回主机的代理软件，代理软件解密后获得所需访问的内容。采用这种技术后，访问请求和访问内容均是经过加密后在网络中传输。

目前，对于采用加密代理技术的软件还没有完善而有效的解决办法。如果能够对这类非法软件进行仔细研究，掌握其内部运行机理、动态更新加密代理的方法以及所使用的协议和算法，就可以有效的对这类软件进行封堵。

综上所述，研究这类软件的加密代理及其动态更新规则，并且对它们所采用的密码协议和算法、加密代理机制、通信机制等进行研究与分析，设计相应的拦截方案至关重要。而要达到此目的，采用反汇编这一手段是恰当而必要的。

## 1.3 研究内容

### 1.3.1 研究目标

本文的研究目标是通过 UltraSurf 软件进行分析，弄清楚它的运行原理，找出可以利用的漏洞，并获得其用于获取加密代理的秘密信息，然后总结出穿透类软件所具有的共性，最后提出针对该类软件的封堵方案。

具体说来，研究目标主要有如下四个：

- 1) 掌握 UltraSurf 软件的运行过程；
- 2) 获得 UltraSurf 软件获取加密代理的秘密信息；
- 3) 总结穿透类软件所具有的共性；
- 4) 设计穿透类软件的封堵方案。

### 1.3.2 研究内容

研究内容有如下五个部分：

#### 1) 脱壳技术的原理与应用

加壳的全称是“可执行程序资源压缩”，是保护文件的常用手段。加“壳”其实是利用特殊的算法，对 EXE、DLL 文件里的资源进行压缩或保护。一般

软件都加“壳”，这样可以保护自己的软件不被破解，加过壳的程序可以直接运行，但是要经过脱壳才可以查看源代码。为了使用反汇编技术分析目标软件，首先要对软件进行脱壳。

## 2) 反汇编技术

众所周知，计算机上的数据是以二进制的形式存放的，可执行文件也不例外。通过使用反汇编工具软件，我们可以将这些二进制数据转换为相应的汇编语言代码，然后在汇编语言的层面上对目标软件进行分析。

## 3) 分析 UltraSurf 软件的内部原理

首先通过脱壳技术对软件进行脱壳，然后使用反汇编工具软件对软件进行分析，得出软件运行的整个过程，从而获得其内部运行原理。

## 4) 获得 UltraSurf 软件获取加密代理的秘密信息

封锁到达特定 IP 的数据包是经常使用的一种网络封锁方法，因此，如果弄清楚软件从何处获得加密代理 IP，便可以使用 IP 封锁技术阻止其获得加密代理，进而封锁这类软件的运行。该研究内容是在分析软件运行原理的基础上进行的。

## 5) 封堵方案的设计

根据分析的结果，设计该类软件的封堵方案。达到减轻或消除这类软件危害的目的。

### 1.3.3 关键技术

#### 1) 软件逆向分析与还原技术

目前，该类基于密码技术的软件的内部结构、运行原理、算法、协议和参数配置均不公开，这些内容直接影响了进一步的信息拦截和内容监视。本文拟用软件反向工程技术对上述软件的运行原理、通信机制、地址配置与更新方式、密码算法、加密代理机制等进行分析与还原，给出具体的交互过程、原始设置、存储格式等信息。

#### 2) 协议与算法分析技术

软件的协议和算法本身可能存在漏洞，这些漏洞或许可以用来阻止软件的



正常运行，从而达到消除该类软件危害的目的。因此，本文将使用协议与算法分析技术对软件的协议和算法进行分析，给出其使用的协议和算法，并且找到其可能存在的漏洞。

### 3) 针对该类软件的信息拦截技术

针对网络通信软件的信息拦截技术有很多，本文将利用现有的技术，结合该类软件的内置信息、运行特点和协议本身可能存在漏洞，制定可以用来阻止软件的正常运行的拦截方案。并在实验室内部署该拦截方案，验证其可行性。

## 1.4 论文结构

本文通过对 UltraSurf8.8 版本(以下如果不做特别说明均简称 UltraSurf)这款典型的穿透类软件进行反汇编分析，获知了它们的运行原理，并提出了针对它们的封堵方案。具体结构如下：第二章介绍了相关的技术背景，包括脱壳技术、软件反汇编技术以及网络扫描与过滤技术等内容。第三章对目标软件进行分析。首先给出了分析 UltraSurf 软件的整个过程，并得出其运行的原理，第四章根据前一章的分析结果，设计了该类软件的封堵方案，并在实验室内对该封堵方案进行了实验。第五章总结了整篇论文，对该类软件的发展及其分析技术进行了展望，并指出了下一步的研究方向。

## 第二章 相关工作

### 2.1 PE 文件格式

PE 文件格式 (Portable Executable File Format) 衍生于早期建立在 VAX/VMS 上的 COFF 文件格式, 目前是微软 Windows 平台上的主流可执行文件格式。EXE 文件和 DLL 文件均使用了相同的 PE 格式, 两者的区别就是用一个字段标识了它们属于 EXE 还是 DLL 文件<sup>[3]</sup>。在对 PE 文件进行脱壳和反汇编前, 必须了解 PE 文件的结构和装入机理, 才能对文件的结构和运行过程进行深入理解。图 2-1 为 PE 文件的主要框架结构。

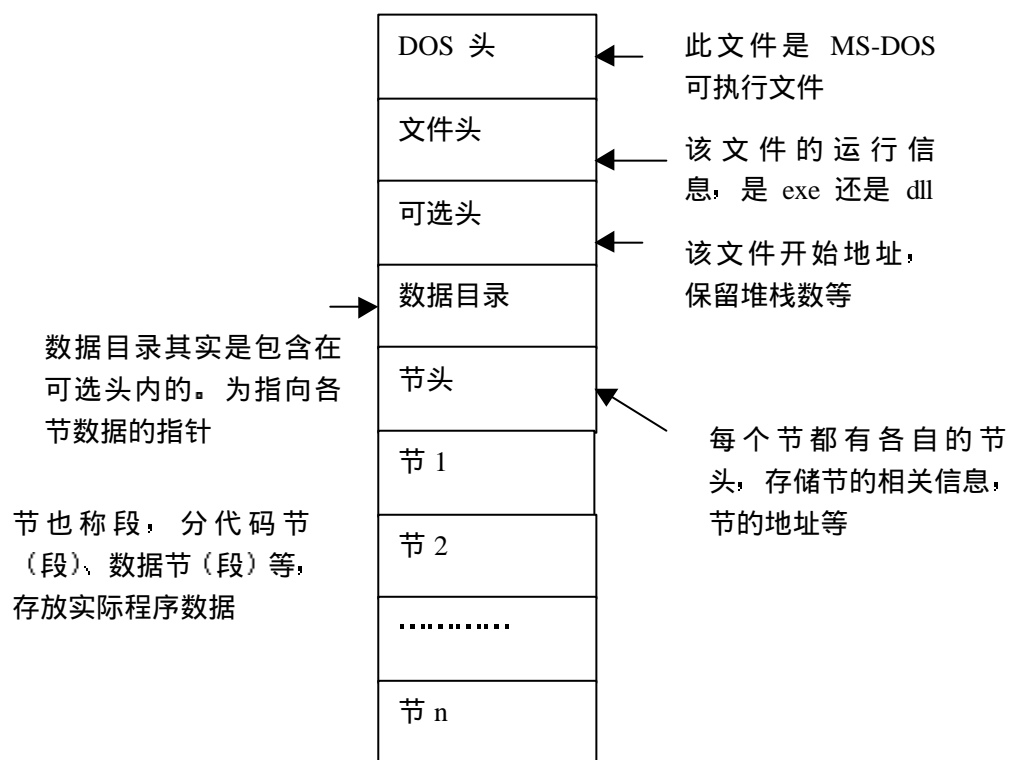


图 2-1 PE 文件框架结构<sup>[3]</sup>

Fig.2-1 PE File Structure

它以一个简单的 DOS MZ HEADER 开始, 它的作用是当程序在 DOS 下运行时, 使 DOS 能识别出它是可执行文件。MZ HEADER 之后的是 DOS STUB,

它的作用是在不支持 PE 文件格式的操作系统中，显示一个错误提示，通常这部分由编译器自动生成，另外该处也可是存放支持 DOS 环境的程序代码。这部分之后是 PE HEADER，它是 PE 相关结构 IMAGE\_NT\_HEADERS 的简称，其中包含了许多 PE 装载器用到的重要信息。执行体在支持 PE 文件结构的操作系统中运行时，PE 装载器将从 DOS MZ HEADER 中找到 PE HEADER 的起始偏移量，直接定位到真正的文件头 PE HEADER。

PE 文件的主体内容划分成块，称之为区块 (SECTION)。每个区块是一段拥有共同属性的数据，比如代码/数据、读/写区域等。值得指出的是，区块的划分是基于各组数据的共同属性，而不是逻辑概念。PE 文件一般至少有两个区块：一个是代码块，一个是数据块。块表 (SECTION TABLE) 是一个结构数组，数组中的每个结构记录了它所关联的区块的位置、长度、属性等信息。

当 PE 文件被执行时，PE 装载器首先检查 DOS MZ HEADER 里的 PE HEADER 偏移量，通过此偏移量直接跳转到 PE HEADER，如果 PE HEADER 有效，就跳转到 PE HEADER 后面的块表，然后 PE 装载器读取块表中区块的信息，并采用文件映射方法将这些区块映射到内存，同时付上块表里所指定的各个区块的属性。PE 文件被映射入内存后，将会从 PE HEADER 里面 AddressOfEntryPoint 结构项所记录的 RVA (相对虚拟地址) 处开始执行。

## 2.2 加壳与脱壳

### 2.2.1 加壳技术

加壳是通过对 EXE、DLL 文件进行压缩和加密来保护文件的一种常用手段。一般软件都通过加壳来保护自己的软件不被破解。本文要进行反汇编分析的这类软件也是经过加壳保护的。

“壳”是一段先于被保护的程序运行的一段程序，在这段程序中对原有程序的代码及数据进行相应的还原解密操作。图 2-2 是被加壳程序的运行过程。

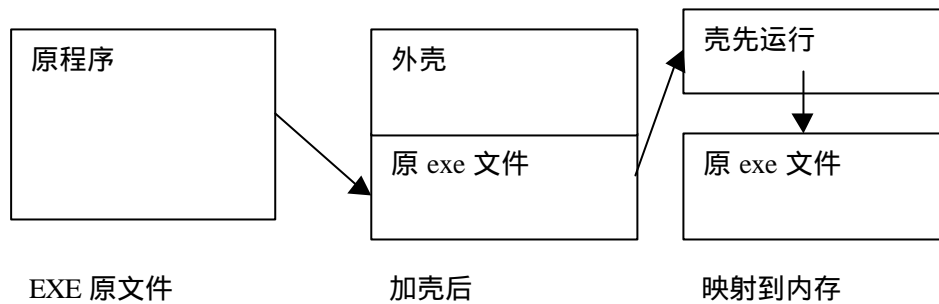
图 2-2 加壳软件的运行过程<sup>[3]</sup>

Fig. 2-2 Execute Process of Shell

按照“壳”的目的和作用，它可以分为两类：一类是压缩壳，另一类是保护壳（又称“加密壳”）。第一类压缩壳主要通过压缩文件的代码和数据来减小程序的体积，加载压缩壳的工具具有 ASPack、UPX 和 PECompact 等。而保护壳则用上了各种反跟踪技术保护程序不被调试、脱壳等，这类壳不考虑加壳后软件的大小，加载保护壳的工具具有 ASProtect、tElock、幻影等。现在，随着加壳技术的发展，这两类壳之间的界限也变得越来越模糊，有些壳在加密软件的同时还对代码和数据进行压缩，这不但减少了文件在磁盘上所占用的空间、有效的保护了软件不被破解，在有些情况下还可能有效的提高了可执行文件的加载速度。

### 2.2.2 脱壳技术

事物都具有两面性，有加壳就必有脱壳。在该文中，脱壳是我们对目标软件进行反汇编分析的前提。

脱壳的方法有两种：一种是借助专用的脱壳工具软件，另外一种手动脱壳。

#### 1) 借助脱壳工具脱壳

在使用脱壳软件脱壳前，需要知道壳的种类和版本号，因为对于不同的软件，甚至不同版本的壳，脱壳处理的方法和工具都不一定相同。目前成熟的查壳种类和版本的工具软件有 FileInfo、PEiD 和 Gtw 等。本文使用 PEiD 来识别壳。PEiD 是一个可以自动完成壳类型及可执行程序的编译器类型的识别工具，目前它能识别大约 400 多种各类加壳软件所加的壳。

脱壳工具软件有很多，常用壳所对应的脱壳工具如下表 2-1 所示。

表 2-1 壳与脱壳工具<sup>[1]</sup>

名称	脱壳工具软件
UPX	UPX 自身 (使用 upx -d 命令), FS, ProcDump
ASPack	AspackDie, CASPR, un-ASPack, DeASPack, Anti-ASPack, ProcDump
Petite	Unpetite, ProcDump
PECompact	PeunCompact, tNO-Peunc, UnPECompact, ProcDump
Neolite	Neolite 自身, ProcDump
PE-PACK	DePEPACK, UnPEPack, ProcDump
ASProtect	AsprStripperXP, CASPR, Asprotect Deprotector, Anti Aspr

## 2) 手动脱壳<sup>[2]</sup>

与借助软件工具脱壳不同, 手动脱壳需要一定的技巧和经验, 其过程一般分为查找程序的真正入口点 (OEP)、抓取内存镜像文件、PE 文件重建三步。

### (1) 查找入口点

入口点是可执行文件被操作系统加载后执行的第一条指令的位置, 也就是记录在 PE 结构中的 AddressOfEntryPoint 这个 DWORD 类型的结构项中的值。壳程序为了先与原来的代码获得执行的控制权, 往往会更改该值。这样, 加壳程序运行时, 首先执行外壳程序, 外壳程序负责把用户原来的程序在内存中解压还原, 并把控制权交给解压缩后的真正程序。查找入口点的任务就是找到程序入口地址。一般说来, 找到一个跨段的 JMP 命令就找到了入口点。当然, 也可以借助入口点查找工具来查找入口点, 这类工具有 D.boy、AsprLoader、PE-Scan、IceDump、TRW2000 等。

### (2) 抓取内存镜像文件

外壳程序解压还原后就会跳到真正的入口点执行, 此时内存映象文件是已解压缩的程序。这时就可以抓取内存映象文件了。抓取内存映象文件可以借助 TRW2000 这个工具, 也可以使用命令法, 不过这需要手动修正 OEP 和块表的偏移地址等。

### (3) 重建输入表

输入表结构中与实际运行相关的主要是 IAT(输入地址表)结构, 这个结果中用于保存 API 的实际地址, 因为壳一般都修改了原程序文件的输入表并自己模拟 Windows 装载器的工作来填充 IAT 中相关数据, 因此我们要根据这个被填充的 IAT 来还原输入表的机构, 这就是输入表的重建。这项工作可以借助 ImportREC 或 Revirgin 等专业的输入表重建工具来完成。

## 2.3 软件逆向工程

软件逆向工程 (Reverse Engineering, 又称反向工程) 是指软件开发过程的逆向过程, 即从可运行的程序系统出发, 对目标文件进行反汇编, 得到其汇编代码, 然后对汇编代码进行理解和分析, 从而得出对应的源程序、系统结构以及相关设计原理和算法思想等。一般将软件逆向工程定义为包含抽取和抽象这两个步骤的过程: 第一步分析目标系统, 标识目标系统的组件以及它们之间相互关系, 第二步创建不同形式或更高抽象层次的系统表示<sup>[5][6][7]</sup>。

随着计算机技术的发展, 软件已经在各个领域普及, 对于软件本身的分析的需要也越来越多。因此对于软件逆向工程的研究越来越引起国内外的重视, 目前已经出现了一些用于软件逆向工程的反汇编工具, 常用的主要有静态反汇编工具 IDA Pro 和 W32dasm, 以及动态反汇编工具 OllyDbg 等。IDA 同 w32dasm 有很多相同的功能: 可以快速到达指定的代码位置; 可以看到跳到指定的位置的跳转的命令位置; 可以看参考字符串; 可以保存静态汇编等<sup>[14][19]</sup>。OllyDbg 是一款动态反汇编工具。它可以预测寄存器内容, 识别进程、API 调用、分支、常量等等, 而且能够动态跟踪其变化。

### 2.3.1 反汇编技术

众所周知, 高级程序语言的指令是无法为计算机所识别, 必须通过编译器进行编译和链接得到二进制代码进行运行。既然计算机只识别二进制文件, 那么, 所有存储在计算机上的文件, 也都是以二进制的形式存放的, 当然也包括可执行文件。计算机只能执行二进制代码, 故我们只能看到可执行文件的二进制代码。从某种意义上来说, 对这些数字和字母进行的组合进行解读几乎是不可能的<sup>[4][21]</sup>。

虽然计算机只能执行二进制代码, 但是二进制的代码和汇编语言之间有一一对应的关系, 可以用一些助记符来代替 0 和 1 的多种组合, 我们通过使用反汇编工具软件可以将这些二进制数值转换为相应的汇编语言代码, 这样可以在汇编语言的层面上对目标软件进行分析。由于反汇编工具很难将汇编代码逆向还原为原始的高级语言代码, 故软件分析一般是停留在汇编语言的层面上。

### 2.3.2 Windows 平台下的反汇编

由于 Windows 的出现,使得可执行文件的执行机制和 16 位 Dos 平台下相比发生了巨大的变化,消息机制的出现,大量 windows API 函数的调用, MFC 和 VCL 等框架的使用使得程序的复杂度急剧增加<sup>[17]</sup>。PE 文件格式的出现和虚拟内存寻址,使得程序的结构和执行流程发生了改变,而异常处理机制的应用使得程序加壳机制得以实现。以上的复杂技术运用,直接导致了 Windows 平台下反汇编难度比传统可执行文件进一步增大,也为研究工作提出了新的命题。

总体来说,基于 Windows 平台的反汇编分析有两种方式,静态分析和动态分析。

所谓的静态分析,就是只通过查看软件的反汇编代码来对软件进行分析。这种分析方式的优点在于可以借助软件工具自动识别出汇编的结构和系统函数调用,减轻人工分析的工作量,缺点是许多动态生成的信息无法查看,对反向分析的信息获取造成了一定的困难<sup>[13][15]</sup>。

动态分析是指在软件运行时,借助调试软件,在操作系统底层直接获取软件运行时的汇编代码。由于软件运行时候信息和行为均可得知,故对于相应的汇编代码易于分析出对应的功能,并且可以通过动态修改程序指令来进行调试。动态分析的缺点是无法在整体上把握软件结构,无法借助软件识别系统函数调用,且调试过程不稳定等。

近年来,以 OllyDbg 为代表的动态静态结合的反汇编工具软件逐渐成为反汇编技术中重要的工具,该软件基于独立开发的 OllyMachine 虚拟机技术,在 OS 平台和应用程序中间插入中间层,使得静态和动态结合的调试成为可能。本文主要采用的就是 OllyDbg 1.10 版进行反汇编分析。

## 第三章 UltraSurf 软件分析

### 3.1 UltraSurf 软件介绍

#### 3.1.1 关于 UltraSurf 软件

UltraSurf 软件是由美国极景网络科技有限公司 (UltraReach Internet Corp.) 推出的一款软件产品, 它是当今最先进的突破网络封锁的软件之一。该软件下载后无需安装和设置, 直接点击就可以启动运行, 一经启动, 用户的电脑屏幕右下角就会出现一个锁头的标识。UltraSurf 将数据加密后传输, 可以帮助互联网用户逃避国家防火墙的关键字过滤, 从而能够自由地浏览任何网站。因此, 通过该软件, 危害所在国家安全和稳定的信息可以通畅无阻地传输到用户本地。

#### 3.1.2 UltraSurf 软件用户界面

双击 UltraSurf 的可执行文件, 出现如图 3-1 所示的对话框, 在“连接”部分状态开始显示为“正在连接到服务器”; 经过 3~4 秒等待后显示“服务器连接成功”字样, 并自动启动 IE 浏览器, 打开显示地址为 <http://www.ultrareach.net/wujie.htm>, 标题为“无界网络”的页面(图 3-2 UltraSurf 信息服务网站首页)。



图 3-1 UltraSurf 启动界面

Fig. 3-1 Start-up of UltraSurf





图 3-2 UltraSurf 信息服务网站首页

Fig. 3-2 Homepage of UltraSurf

点击“退出”，会弹出“退出警告”（见图 3-3），20 秒后 IE 浏览器关闭。

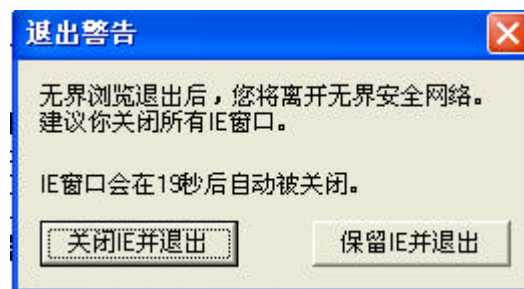


图 3-3 UltraSurf 软件退出警告

Fig. 3-3 warning of UltraSurf when exit

关闭 IE 浏览器后，再访问 <http://www.ultrareach.net/wujie.htm> 则显示“找不到服务器”。若选择不关闭 IE 浏览器，还是显示找不到服务器。由此推断，这个网址的 DNS 解析是由运行软件后的某种机制来完成的。进一步验证：如果在 windows 窗口下运行 cmd 打开命令窗口，然后用 nslookup 命令(查询 IP 和域名的对应关系)也无法获得 <http://www.ultrareach.net/wujie.htm> 的实际 IP。这个发现为我们以后反汇编分析指出了某些参考方向。

### 3.1.3 UltraSurf 软件特点

作为互联网上最流行的突破网络封锁的软件之一，UltraSurf 不仅有着体积小，连接服务器和穿透封锁自动化等优点，而且通过使用加密代理，动态

地址搜索等技术使得经过这些软件处理的信息难以辨别和封堵。UltraSurf 特别支持 Google 搜索引擎，使搜索到的内容不被过滤。

该软件运行时可以选择自动搜索代理和手动配置代理，软件成功运行后，会自动打开 UltraSurf 的主页，该主页的地址一般是本机软件代理所使用的本机地址，使用的端口是 9666 (<http://127.0.0.1:9666/>)。该软件也可以在使用的过程中根据用户的需要重新搜索代理服务器，并可在软件关闭的时候删除 Cookie 和浏览器的历史记录。

该软件较新版本实现了透明、实时的网络加密传输，支持几乎所有基于 HTTP 的功能，而且支持用户上传数据。

## 3.2 UltraSurf 软件分析与脱壳

基于反汇编分析的困难性，在对软件做反汇编分析前，需要完成前期准备工作，以便反汇编工作可以顺利进行。这些准备工作包括软件信息分析，软件行为监测和软件脱壳等。

### 3.2.1 软件信息分析

软件信息分析包括软件开发与编译工具的分析，软件加密方式的分析和软件所加壳的类型的分析。

了解软件的开发工具，可以预先判断软件所使用的库函数和相应的执行方式，在反汇编工作中可以使用相应的工具整理出软件调用的库函数，对于此类函数不必进行深入的跟踪，只需分析其调用的参数和实现的功能即可。同时，了解开发工具可以帮助反汇编分析更好的正向思维，还可以利用开发工具编写相同功能的程序进行比较。

了解生成软件所使用的编译器，可以通过掌握该编译器对高级语言特殊的编译和优化方式，以利于反汇编分析时识别特定的软件结构。

了解软件对可执行程序的保护方式是软件信息分析中非常重要的一环，当前许多软件为了保护自己的软件信息不被反向分析，都采用了加壳技术，如果不了解软件所加壳的种类就贸然进行分析，必将导致盲目而无效的分析结果，所以在软件分析时先了解软件的保护方式，必要时进行脱壳处理，这是非常重要的一项工作。

针对 UltraSurf 软件的信息分析如下：

使用 PEiD 查看 UltraSurf 软件的信息如图 3-3 所示。

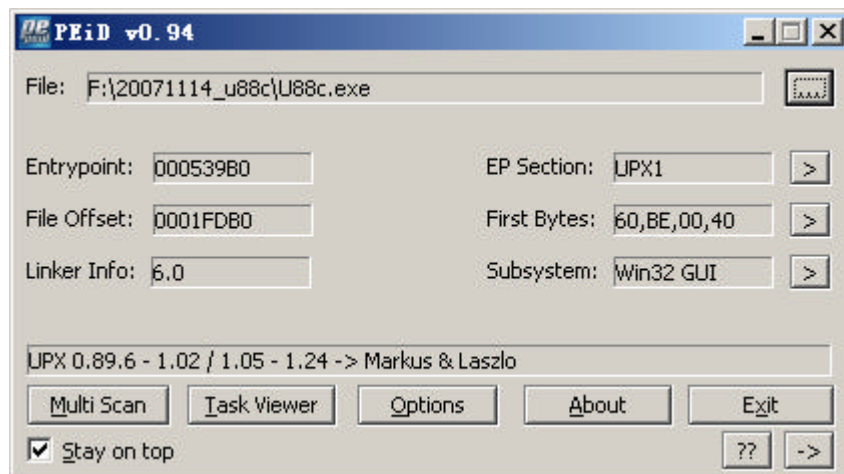


图 3-3 UltraSurf 软件信息

Fig. 3-3 The Information of UltraSurf

由此可知，该软件的保护壳为 UPX 格式。UPX 是一种常用而且著名的压缩加壳程序，这对于我们的脱壳工作提供了重要信息。

### 3.2.2 软件行为监测

在 Windows 平台下执行的应用程序不是孤立的执行的，它一般要调用大量的 Windows API 函数和库函数，同时要建立和读写文件，修改注册表等等，而本文所分析的软件则必然涉及到网络信息的发送、接收、收集和整理等行为，使用黑箱分析方法，可以在没有使用反汇编技术之前了解软件的整体运行情况和流程，这对反汇编期间程序行为的理解有非常重要的帮助。

在软件反汇编的动态运行期间，使用系统监视工具同样可以检测软件行为，从而利于反汇编的分析。例如，用网络监视工具可以帮助定位网络连接的所对应的汇编代码，使用文件监视工具可以定位文件读写所对应的汇编代码。

UltraSurf 软件成功运行后，会自动启动 IE 浏览器，并打开端口为 9666 的本机地址 (http://127.0.0.1:9666)，因此我们推测该地址是代理服务的地址，为了验证该猜测，我们查看 IE 浏览器的设置，可以发现，UltraSurf 软件会在启动后修改系统的 Internet 配置，设置一个虚拟的代理服务器地址，如图 3-4 所示。

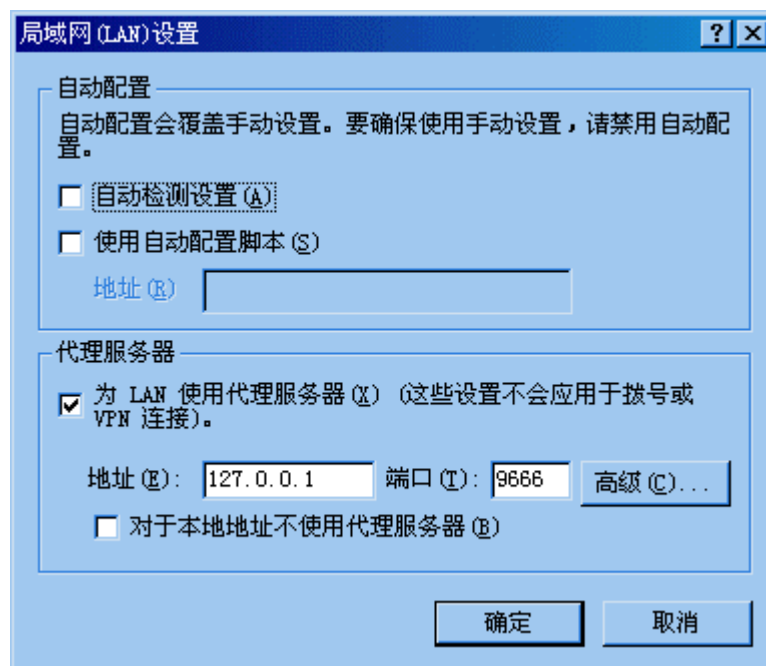


图 3-4 UltraSurf 对网络设置的更改

Fig. 3-4 The Changed LAN Setting

其次，我们使用文件监视工具分析 UltraSurf 软件的文件读写行为，可以发现，UltraSurf 软件主要读取两个临时文件，同时还会使用 cookie，这些操作的具体意义我们暂时无法得知，但可以记录下来，留待后面反汇编阶段进行深入分析。

### 3.2.3 软件脱壳

软件脱壳与软件执行加密密切相关，绝大多数的情况下，Win 32 应用程序的保护方式都是通过加壳技术完成的。关于加壳与脱壳的相关技术前文已经有所阐述，如果针对没有脱壳的软件，使用反汇编调试工具进行分析，得到的将会是无意义的汇编代码，执行流程将会完全不符合正常逻辑。因此必须进行脱壳后方可进行正常的反汇编工作。

由前面的信息分析我们得知 UltraSurf 所加的是 UPX 壳，另外，它脱壳前的原始版本大小为 106K。由于 UPX 程序对加壳程序提供了解压缩功能，我们使用 UPX 程序将软件脱壳。即可得到未加密的程序版本。使用 UPX 脱壳的命令是：

```
upx -d u88c.exe
```

脱壳后，再次使用工具软件 PEiD 可以看出程序是由 VC++ 6.0 编译得到的。如图 3-5 所示。

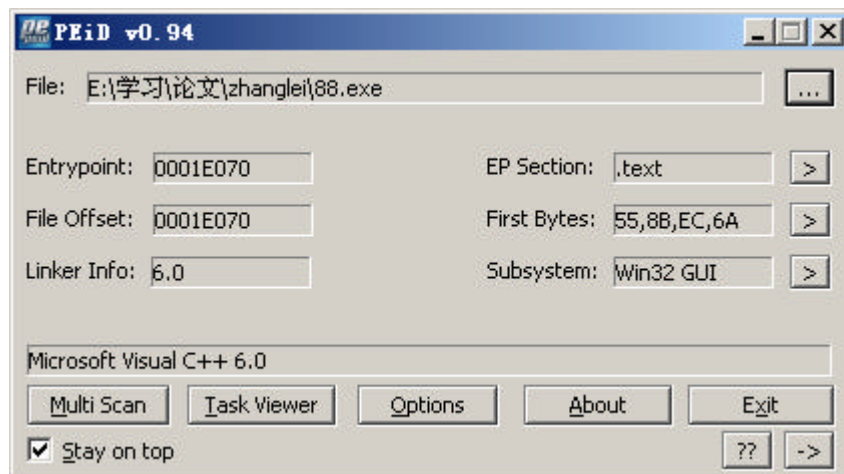


图 3-5 UltraSurf 软件脱壳后信息

Fig. 3-5 The Information of Unpacked UltraSurf

这样，我们就可以根据前面得到的信息，通过相关的反汇编分析工具对 UltraSurf 软件进行反汇编分析了。

### 3.3 UltraSurf 软件反汇编分析

#### 3.3.1 OllyDbg 介绍

##### 1) OllyDbg 的安装与配置<sup>[10]</sup>

OllyDbg 1.10 版的发布版本是个 ZIP 压缩包，只要解压到一个目录下，运行 OllyDbg.exe 就可以了。汉化版的发布版本是个 RAR 压缩包，同样只需解压到一个目录下运行 OllyDbg.exe 即可。OllyDbg 启动后各个窗口的功能如图 3-6 所示。

下面简单解释一下各个窗口的功能：

##### (1) 反汇编窗口：

显示被调试程序的反汇编代码，标题栏上的地址、HEX 数据、反汇编、注释可以通过在窗口中右击出现的菜单 界面选项->隐藏标题 或 显示标题 来进行切换是否显示。用鼠标左键点击注释标签可以切换注释显示的方式。

##### (2) 寄存器窗口：

显示当前所选线程的 CPU 寄存器内容。同样点击标签寄存器可以切换显示寄存器的方式。其中包括通用寄存器 EAX、EBX、ECX、EDX

和 ESP、EBP、ESI、EDI 以及 EIP 等寄存器。

(3) 信息窗口：

显示反汇编窗口中选中的第一个命令的参数及一些跳转目标地址、字符串等。

(4) 数据窗口：

显示内存或文件的内容。右键菜单可用于切换显示方式。

(5) 堆栈窗口：

显示当前线程的堆栈，里面的数据对于分析线程的数据变化非常有用。

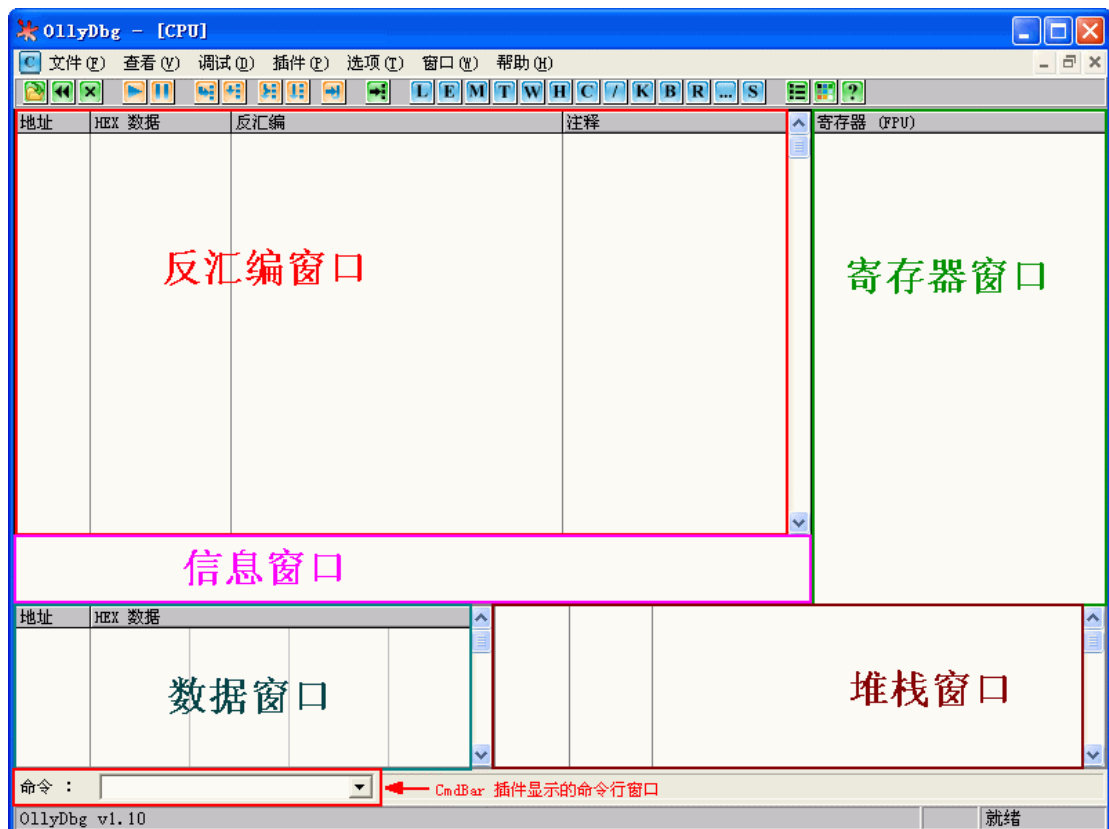


图 3-6 OllyDbg 调试窗口

Fig. 3-6 The Debugging Window of OllyDbg

## 2) OllyDbg 的使用

OllyDbg 有三种方式来载入程序进行调试，一种是点击菜单 文件->打开来 打开一个可执行文件进行调试，另一种是点击菜单 文件->附加 来附加到一个 已运行的进程上进行调试。注意这里要附加的程序必须已运行。第三种就是用

右键菜单来载入程序。一般情况下我们选第一种方式。

调试中我们经常要用到的快捷键有：

F2：设置断点，只要在光标定位的位置（上图中灰色条）按 F2 键即可，再按一次 F2 键则会删除断点。

F8：单步步过。每按一次这个键执行一条反汇编窗口中的一条指令，遇到 CALL 等子程序不进入其代码。

F7：单步步入。功能同单步步过(F8)类似，区别是遇到 CALL 等子程序时会进入其中，进入后首先会停留在子程序的第一条指令上。

F4：运行到选定位置。作用就是直接运行到光标所在位置处暂停。

F9：运行。如果没有设置断点的话，按下此键被调试的程序将直接开始运行。

CTR+F9：执行到返回。此命令在执行到一个 ret (返回指令)指令时暂停，常用于从系统领空返回到我们调试的程序领空。

ALT+F9：执行到用户代码。可用于从系统快速返回到我们调试的程序。



### 3.3.2 通过分析字符串猜测软件内部信息

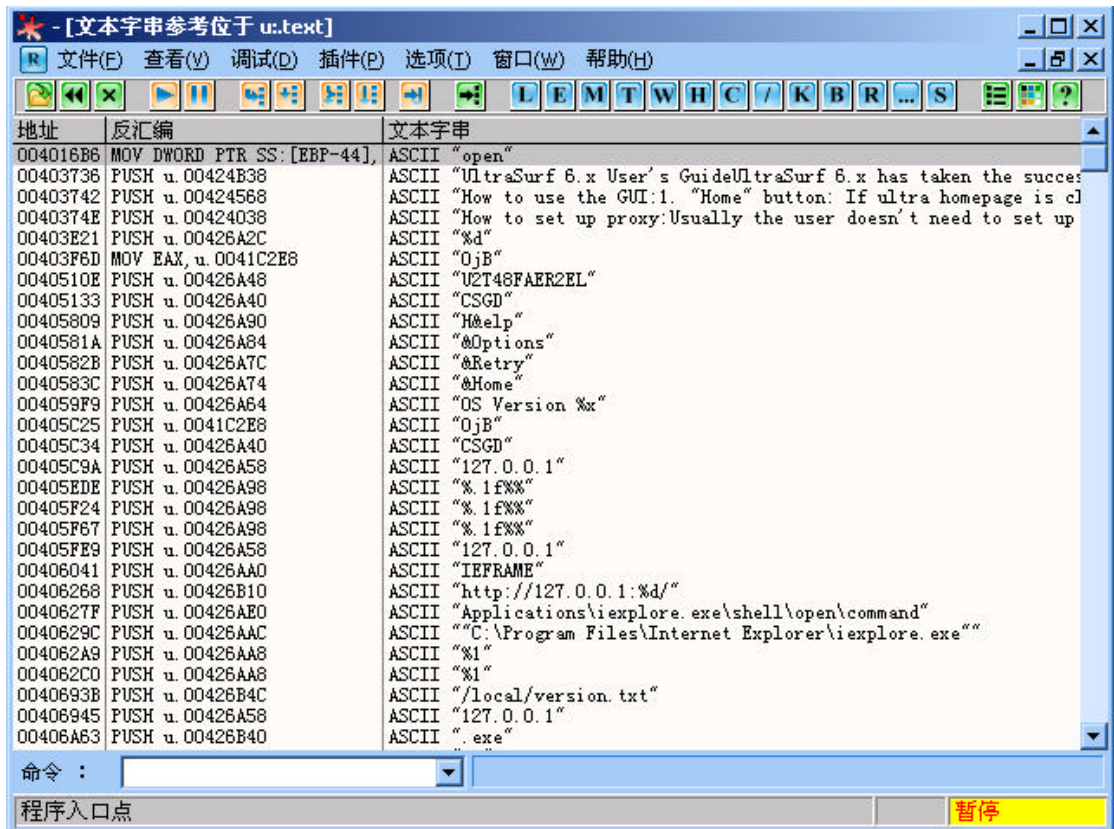


图 3-7 字符串信息

Fig. 3-7 Character String Information

在对软件的整体结构和分析思路还没有把握的时候，由于软件内部的字符串可以通过工具查看到，首先从这里开始寻找一些有用的信息。

用 OllyDbg 载入软件，在反汇编代码使用“查找-所有参考文本字符串”，可以看到如图 3-7 所示字符串信息。

从中我们不难得出程序的使用信息，调用系统参数和系统程序，以及连接 IP 的信息。我们可以通过在这些字符串对应的地址附近查看对应的汇编代码来寻找相应的功能函数。

### 3.3.3 从程序入口点到实际函数调用

程序入口点是可执行文件被操作系统加载后第一条指令的位置。在使用调试器加载程序时，程序载入后停在的第一条指令的地址就是程序的入口点。入口点一般在 0x400000 的地址附近，实际的入口点地址根据实际需要而定。



入口点地址在脱壳以及验证脱壳的正确性时比较有用，在获得脱壳后的软件的汇编代码后，通常从程序入口点到我们所感兴趣的代码段还需要一段时间。

按 F8 使程序单步运行。在单步步过一个 CALL 时（CALL 代表一个函数），弹出 UltraSurf 的主界面，程序开始运行。观察此 CALL 所在的地址附近的代码，发现此代码前面调用了 GetStartupInfo 和 GetModuleHandle 两个函数，后面调用了 exit 函数，根据编程经验和 MSDN 得到，前两个函数都是在做程序运行前的准备工作，exit 是程序运行完退出。所以我们在运行到中间的函数时可以按 F7 跳入继续分析，在跳入后的第一条指令上按 F2 设置断点，可以在加载完程序后直接按 F9 跳到有断点的指令处。

一般程序载入点是 0x400000<sup>[12]</sup>。所以在 0x400000 附近一般可以找到我们需要重点分析的代码，尽管在执行该段代码之前，若是 MFC 程序，则需执行很多 MFC42.dll 以及 USER32.dll（用于界面）的函数，但是都会从 DLL 中通过跳转的方式（JMP 指令或 CALL 指令）回到主程序代码段中。

需要说明的是，一个程序经过反汇编和 OllyDbg 的加载，可能生成非常多的汇编代码，如果算上调用的 USER32, KERNEL32 等 DLL 模块的代码，数量就更多。对于反汇编者来说，对一个程序的反汇编必定有某种目的或者需要了解程序运行的流程，所以在反汇编的过程中需要懂得研究对自己有意义的代码和跳过无意义的代码（比如仅仅是起初始化界面的作用），从程序入口点到实际函数调用的分析就是这样一个对代码进行初步分析的最简单的例子。

### 3.3.4 MFC 程序的调试

进入上述的函数调用以后，我们发现了下列语句：

```
004173F6|.      E8      43000000      CALL
<JMP.&MFC42.#1576_?AfxWinMain@@YGHP>
```

从 MFC 程序的知识来看，说明 UltraSurf 软件是一个利用微软的 MFC 技术编写的程序。

MFC 是微软提供的一个应用程序框架，它包装了 Windows 的消息机制。从程序编写者的角度而言，MFC 技术简化了程序编写者记忆 Windows API 的繁琐过程，简化了在 Windows 操作系统下创建 C++ 应用程序的过程，但是从

程序反汇编调试的角度而言, MFC 使得 Windows API 上又多了一层包装, 对反汇编者的要求是和调试单纯的使用 Windows API 的程序不一样的<sup>[9]</sup>。

利用反汇编方法调试 MFC 编写的程序的特点是程序本身需要使用的代码地址都被压入堆栈中, 通过出栈操作可以重新找到这些地址, 然后可以从 MFC 中通过对这些地址的调用重新切入到程序模块。

继续跟进 AfxWinMain 中分析, 在一段 MFC42.dll 的代码后发现如下代码:

```
0040538E      .E8      CF190100      CALL
<JMP.&MFC42.#2514_?DoModal@CDialog@>; // MFC 程序运行循环函数
```

由 MSDN 知, 此函数为 MFC 的循环函数, 创建有模式的对话框, 而单步步过此句时 UltraSurf 软件完成整个运行过程, 可以推断 UltraSurf 软件的界面对话框是通过 MFC 的 DoMal()函数创建的。

图 3-8 是关于程序体从 MFC 代码段返回用户代码段出栈的实例:

执行到 73D3CF6D 地址, 可以看到对于 EAX 寄存器的操作, 之后在 73D3CF71 地址上, 对 EAX 加上偏移地址量 58, 得到的是应用程序的代码地址 004050D4, 这样就返回到应用程序模块中去了<sup>[11][20][22]</sup>。

73D3CF63	FF90 8C000000	CALL DWORD PTR DS:[EAX+8C]	
73D3CF69	85C0	TEST EAX, EAX	
73D3CF6B	74 2A	JE SHORT MFC42.73D3CF97	
73D3CF6D	8B06	MOV EAX, DWORD PTR DS:[ESI]	
73D3CF6F	8BCE	MOV ECX, ESI	
73D3CF71	FF50 58	CALL DWORD PTR DS:[EAX+58]	u. 004050D4

图 3-8 MFC 返回应用程序

Fig. 3-8 Return From MFC

### 3.3.5 多线程分析

在调试中, 发现很多代码被连续并反复调用。开始猜测同样的代码运行数次是因为迷惑调试者, 但后来发现很多 AfxBeginThread 的函数存在代码中, 共十多处。所以猜测同样的代码运行多次是因为它们在不同的线程中运行。调试器同样提供了查看当前线程的功能, 虽然不能准确预知某时刻究竟是哪个线程在运行, 但是可以分析出发生了上下文转换。

我们利用调试器的功能找到所有的 AfxBeginThread 并在上面下断点分析。由 MSDN 知, AfxBeginThread 有两种形式, 一种称为“worker”, 一种称为“UI”, 前者负责进行运算, 后者负责进行界面处理。

UltraSurf 软件中共调用了 9 次 worker 形式的 AfxBeginThread 和 1 次 UI 形式的 AfxBeginThread。UI 并非我们关注的重心，所以分析的重点放在“worker”形式的线程。

9 个线程的作用经初步静态分析如下：（线程编号顺序按其在内存中的地址）

线程 A:与无界主页连接，下载最新版本号等

线程 B:负责 UDP 连接

线程 C:做连接之前的准备工作，和界面有关的操作等。

线程 D:和写入发送信息有关

线程 E:和发送信息有关

线程 F:软件利用 443 端口与加密代理的通信

线程 G:处理收到的回应

线程 H:和本地代理服务器设置有关

线程 I:测试软件对外连接是否正常

由于在运行中还需要利用动态分析技术分析各线程的作用，完整的流程以及这些线程具体的使用还会在后续分析中进一步描述。

### 3.3.6 UltraSurf 软件的加密与解密

#### 1) 临时文件的生成

继续分析时，调试器中分析出的 fopen, fwrite, fread, fseek, ftell 等函数引起了注意。

这些函数都是与文件读写相关的函数。可以用 Filemon 软件来查看系统中临时文件夹中文件写入的情况，发现完整运行一次软件后，在系统临时文件夹中写入一个 8 位文件名的文件，用十六进制编辑器打开后是无法辨认内容，怀疑这段内容被加密，而在程序中读入这个文件并解密。

利用反汇编手段验证，在调试过程中，发现某段时间寄存器被写入一个 ASCII 字符串：“C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\”其中“ADMINI~1”为当前使用机器的用户名。由此推断完整的字符串应该为“C:\Document and Settings\当前用户名\Locals and Settings\Temp”，将 Filemon 的过滤器设为此文件夹，关闭其他可能造成干扰的程序，再次从头运行，发现有三个新文件被生成，但是完整运行软件一次后只有其中一个文件留在文件夹内。

这三个新文件的相同之处是文件名都为 8 位，用十六进制编辑器打开后都为不能识别的乱码，怀疑此为采用某种加密算法加密的数据。

在调试器内运行程序到一半，然后终止运行，发现三个文件都还存在，而完整运行后其中两个就被删除。经这样的对比后确定当前分析目标是：找出写入文件及读取，生成文件名的算法，以及删除文件的算法，并且了解删除临时文件的原因。

OllyDbg 提供了一个很有用的功能，即根据函数名称找出程序代码中所有调用此函数的地方。利用这个功能我们可以得到调用 fopen, fread, ftell, fwrite 等函数的列表，然后选择“在所有参考上下断点”，就会在运行到这些函数时停止，方便我们的跟踪。

重新载入运行，在第一个 fopen 函数附近单步跟踪分析。在运行 fopen 函数之前，程序运行一连串的函数，如下：

```
00405C31 . E8 E4720000 CALL u.0040CF1A
00405C36 . E8 2B750000 CALL u.0040D166
00405C3B . E8 4E710000 CALL u.0040CD8E
00405C40 . E8 07740000 CALL u.0040D04C
00405C45 . E8 5C760000 CALL u.0040D2A6
```

单步跟过以后文件便存在于临时文件夹中。所以对此段代码进行深入分析。

在第一个 CALL 的代码中包含了第一个 fopen。在前面调用另外一个 CALL。所以推断这个 CALL 函数生成要打开的文件名。

跟进后，发现程序调用了 GetTempPath 此函数在寄存器 EBX 中写入前述的 ASCII “C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\” 紧跟一个 CALL。继续跟进，经过一系列压栈操作后，调用一个 CALL。此 CALL 函数包含 GetWindowsDirectory 和 GetVolumeInformation 两个函数，并且可见 GetVolumeInformation 包含一个 “C:\Windows” 的参数。查找 MSDN，GetWindowsDirectory 的作用为得到 Windows 所在目录的路径，这也就是下面 “C:\Windows” 的来源。而 GetVolumeInformation 获得指定盘符的序列号。我们也可以通过在开始->运行->cmd 窗口，在定位到指定盘符后输入 “vol” 命令看到该盘盘符(如图 3-9 所示)。

在调试器里运行完含上述两函数的 CALL 后发现寄存器 EAX 改变为 “C05F0611”，用 vol 命令看到的 C 盘卷序列号同样为此数值。

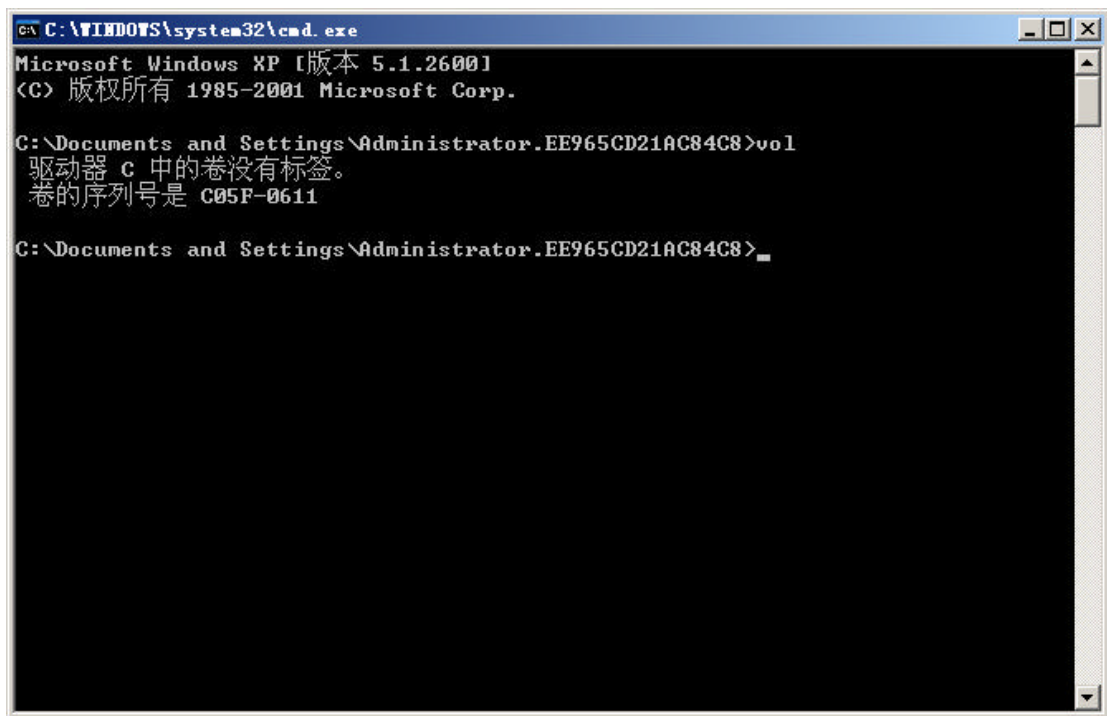


图 3-9 利用 vol 命令获得磁盘卷序列号

Fig. 3-9 Get Disk serial by vol

所以生成文件名的算法应该与机器 windows 所在目录的卷序列号有关。在别的机器上运行同样的程序生成不同文件名的试验验证了这一点。

所以生成文件名的算法因机器而异，一机一码。

继续分析下面的代码，看到很多 ADD, DIV, SHR, XOR 等改变数值的指令，推断此部分是生成文件名的代码，仔细分析得到生成文件名的代码，详细汇编和 C++ 代码见后。

此处用到了在 CALL 一开始压入堆栈的数值，推断此为函数传入的参数。

下面一个 CALL 调用了相同的过程，但是使用到的参数不同，生成了另外一个文件名。下称这两个文件分别为文件 1，文件 2，这两个文件为软件在完整运行后删除的文件。另外一个临时文件下称文件 3，软件完整运行后不删除。

生成文件名（8 位）后，调用 strcat 函数将前面获取的临时文件夹路径与文件名连接起来，即得到临时文件完整的路径。

连续的四个 CALL 调用，已经分析出前 2 个是为了生成文件名，后两个是写入数据。也可以通过查看文件内容得到此结论，前两个 CALL 运行完以后文件内容为 0，而后两个 CALL 运行完以后文件开始有不可辨别的内容。

除了生成上述所说两个文件名外，此算法还生成了一个注册表中的字符串名，在注册表文件的生成这部分将进一步介绍。

此部分只生成了两个程序运行结束后会删除的文件，另外一个文件是在何时生成？先猜测此文件名也是用同样的算法产生的，在此算法上下断点，每次运行都检查其生成的文件名并比较。

在跟踪运行时，发现在 UltraSurf 软件的界面出现之后又中断在此生成文件名的函数处，单步运行跟踪时，发现此时生成了文件 3 的文件名。

此处有一个前提条件没有提及：在调试运行软件时，文件 3 已经存在于临时文件夹中。如果临时文件夹里并不存在该文件，生成临时文件名的过程会不会有所改变？

根据此疑问，删除临时文件夹中的文件 3，重新在调试器中运行软件。生成文件名的算法调用的位置还是一样，但是生成文件名后的一条跳转指令似乎值有所改变。推测此处为判定文件名是否存在的算法，如果没有该文件则生成，如果有该文件则读取。

想知道该文件中的内容，猜测需要寻找一个解密算法来将文件中的内容转化为有实际作用的数据。

文件 1 和文件 2 运行结束就被删除，似乎是没有用的文件，利用黑箱分析方法验证这一猜想，在调试器中载入文件运行后，在文件 1 和文件 2 生成后，暂停软件运行，在临时文件夹中删除它们，然后继续运行软件。

运行结果是软件显示“请检查网络连接”，中止了运行。而软件实际上并没有检查网络连接是否正常。

由此推测，文件 1 和文件 2 与检查网络连接有关。而且观察到每次运行软件后，文件 1 和文件 2 的内容不同，猜测与一些必须更新的信息有关。

## 2)注册表文件的生成

将断点下在文件名生成的函数时，在生成文件 2 以后，会生成另外一个 8 位文件名，但这个文件名并不写入临时文件夹。在附近代码进一步分析，发现如下与注册表有关的函数：RegOpenKey,RegQueryValue,RegCloseKey. 继续分析发现此时生成的文件名是注册表中的键值名，位置位于：`\HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\Settings`。这是一个存储对 IE 的设置的注册表位置，怀疑这和 IE 的设置有关，但是在调试器中运行到此处时，在 IE 的局域网代理里并没有设置代理，所以这个注册表键值和 127.0.0.1 的代理并没有关系。

在调试器中完整运行了一次软件，打开无界主页后，点击 UltraSurf 对话框上的各按钮，进行黑箱测试，在点击“高级设置”按钮后，调试器中的软件在生成文件名的函数上被断下。继续跟踪，发现该函数生成的文件名正是上述

文件名，而此时弹出了 UltraSurf 软件高级设置的对话框，包含是否在程序启动时自动启动 IE、是否在程序退出后删除 cookie 等选项。

由此结合白箱分析的结果推断，所分析的 UltraSurf 软件的配置信息是通过此键值存储在注册表中的。这个键值也是根据机器 Windows 所在文件夹的磁盘序列号计算出来的，只是使用了与生成文件名不同的参数。软件运行的开始会读取这个配置。

### 3)加密和解密数据

UltraSurf 软件通过加密解密来突破关键字过滤的封锁，所以研究 UltraSurf 软件的加解密方式和算法十分重要。在软件运行过程中，临时文件和注册表中保存的，程序中固定的，网络上传送的信息均需要经过不同的加解密过程。

重点分析读取文件 3 后运行的，修改内存的一段代码，此段代码在程序的多个位置被调用，怀疑它写入内存的是解密后的数据。

在此段代码上下断点，重点观察写入内存的内容。开始两次调用此函数时，让内存的某一段全部为 0 和全部为 04000000 的重复。

在文件 3 不存在于临时文件夹中时，此段代码第三次被调用时生成了几十个 IP 并写入内存，以后几次生成同样 IP。而这段内存存在我们开始运行程序时只是一堆杂乱无章的 ASCII。将这些生成的 IP 保存以便于以后的分析。

继续运行并观察此函数的运行情况，在写入内存的指令上下断点以方便分析，在随后的运行过程中发现此函数在与第三次被调用时相同的内存位置写入了另外数十个 IP，也保存下来以便分析。

相同的运行过程持续了 5-8 次。

此函数的最后一次调用在内存中写入了 40 多个连续的 IP，与前面的 IP 都不同，同样重复几次。把这些 IP 也保存下来等待分析。

结合这些 IP 分析，我们可以用网络连接监控的方法继续分析软件。

同时也猜测还有其他的加解密函数的存在。

在点击“退出”退出软件时，发现仍会调用此段代码，将部分内存写入为“0000”和“040000”似乎此段代码还有初始化内存的作用。

此段代码的详细分析见 3.5.2 节。

运行 C++ 写成的此段代码，读入文件 3，可以得到一连串的 IP。一般是 4-9 个，数目不定，而且每次软件运行后，文件 3 的内容会不同。

结合后续分析表明，此段代码可以用作 DNS 服务器 IP、代理服务器 IP、以及临时文件中 IP 的密文至明文的转换过程。

### 3.3.7 UltraSurf 软件的网络连接

#### 1) 网络连接初步分析

UltraSurf 软件是一个互联网软件，所以它的运行与网络条件密切相关。要分析软件的行为，首先要对软件运行的网络环境有所了解，并能对其使用的网络协议，连接到的 IP 等给出分析。

在本次研究中我们使用网络抓包软件 Ethereal 和 Outpost 防火墙软件结合的方法来对软件运行时网络使用情况作初步黑箱分析。

利用 Ethereal，可以在关闭其他所有的互联网软件的前提下分析软件运行时使用了哪些类型的连接，和哪些 IP 进行通信。

在文件 3 存在和不存在于临时文件夹两种情况下分别进行抓包分析，得到了不同的结果。

文件 3 存在于临时文件夹中时：

抓包结果中只有 TCP 连接和 SSL 连接，与多个 IP 通信，开始的 IP 都不在已经分析出的 IP 列表中；后面有部分 IP 存在列表中；

文件 3 不存在于临时文件夹中时：

抓包结果除了 TCP 和 SSL 连接外，增加了 DNS 请求，而且 DNS 请求一些很奇怪的域名，例如：ns2.d79872fb4.net，并且从某些 IP 地址返回了对 DNS 请求的回应。TCP 和 SSL 连接的结果与文件 3 存在时近似，但是连接到的 IP 几乎不相同。将返回 DNS 请求回应的 IP 地址视为 DNS 服务器，它们也不在已知的 IP 列表当中。在抓包结果中查找 DNS 服务器返回的 IP 地址，未找到相对应的地址。

开启 Outpost 防火墙可以询问用户是否允许软件连接某一个 IP。这样就能够控制软件的网络连接，从而人为制造某些连接是否成功的情况。

可以推断，文件 3 中的内容与连接有关。在文件 3 存在时，软件读取文件 3 中的内容并解密，然后连接。在文件 3 不存在时，软件需要发送一些 DNS 请求到某些 DNS 服务器，然后再利用返回的 IP（可能并不是直接利用，而是利用某种算法转换）。

软件存在尝试网络连接是否正常的过程。因为每次软件都连接到一些无规律也不属于解密出的 IP 的 IP 地址，如果一直阻止其连接，它就会显示“网络连接失败”而不进行后面的连接过程。所以初步的对网络连接分析得到了一个猜想：是否可以通过干扰软件尝试连接的过程来封堵软件。

由前面的分析结果得到，在软件运行中途删除文件 1 和文件 2 时，软件无



法判断网络连接是否正常。结合此结果判断，文件 1 和文件 2 似乎是写入了临时使用的 IP，所以在它们不存在时，软件无法取得随机 IP，所以显示网络连接不正常。所以，在每次软件退出后都要删除文件 1 和 2，以防止每次测试的 IP 都一样，否则就可以利用这个漏洞被封堵。又利用白箱方法证明了这样的过程：在软件运行的开始如果检测到已经有文件 1 或 2 存在时，软件会先删除这些文件，然后再生成新的文件，利用随机数生成 IP，加密并写入文件。

由于不可能在客户端上运行自动删除文件 1 和文件 2 的程序，所以尽管只要随时删除临时文件夹内的文件 1 和 2 就能使 UltraSurf 软件无法使用，还是要寻求别的外部封堵方法。

## 2)DNS 服务分析

UltraSurf 软件中存在解密 DNS 服务器 IP，解密 DNS 域名和向 DNS 服务器发送域名的过程。在汇编代码中寻找这个过程。

在文件 3 不存在于临时文件夹的情况下：

在一段连续的内存地址内存放类似“ql.ly.~{z,\*{1qzk”的代码，经跟踪发现经一段循环代码后，此处内存地址被写入类似“ns1.flade735d.net”的内容，约 40 余个，判定此处为解密 DNS 域名字符串处。

继续跟踪寻找这段域名字符串被读取的代码。在尝试网络是否正常的代码后，发现与此段 DNS 字符串有关的代码，每次读取一个形如 ns1.flade735d.net 的字符串，进行多线程连接向 DNS 服务器发送域名，并且发现下列形式的字符串：

```
May-21-13:50:05 | 929453: Send IDURL Query ns1.062efa01c.net to node 71.229.238.191.
```

查找这个 IP，发现在我们以前通过解密函数找到的 IP 中，所以初步断定这个 IP 代表了一个 DNS 服务器的地址，UltraSurf 软件是通过向内置 DNS 服务器发送特定的域名请求来达到动态域名解析的目的的，一个域名可以有多个 IP 地址，所以这是对 UltraSurf 软件封堵的一个难点。

利用 Etheral 抓包的结果证实了我们的推测，有包含此形式的 DNS 请求发出，并有服务器返回 DNS 查询结果，而且不止一个结果。

在文件 3 存在于临时文件夹内的情况下：

在文件 3 存在于临时文件夹，也就是有与连接相关的 IP 的情况下，对 DNS 相关代码进行分析。可见，第一段解密 DNS 域名字符串依然按上述方式工作，第二段则直接跳过了 DNS 查询的代码而直接开始连接代理服务器。

我们知道，网络环境多变，前一次建立了连接并不能保证后一次也能建立

连接。所以我们进行进一步的在网络环境进行控制的测试。

在文件 3 存在于临时文件夹中的情况下，打开 Outpost 防火墙，利用已经找到的解密算法，将文件 3 中的 IP 解密出来，利用 Outpost 的拦截功能，拦下文件 3 中包含的所有 IP。然后在调试器中我们看到，软件又跳到了 DNS 查询的代码指令并且开始查询。所以推测软件找寻连接的 IP 经此步骤：检查文件 3 看是否存在->若不存在则进行 DNS 查询，若存在则先连接到文件 3 中的 IP，若文件 3 中的 IP 无法连接，则继续进行 DNS 查询来寻找连接的 IP。

又观察到，DNS 查询的 IP 并非查询后直接连接的 IP。所以推测存在一个算法来将返回的 IP 转换成实际的 IP。反汇编分析证明了这个猜测。

若 DNS 查询出来的 IP 也无法连接呢？我们又采用拦截功能将转换出的 IP 拦截下。

软件仍旧在继续连接，但是连接的 IP 似乎是以前通过解密出来的 IP。所以推测解密出来的长串 IP 为内置代理服务其 IP，在两种连接方法都失效以后采用这种连接手段。当继续拦截这些 IP 时，软件最终宣告连接服务失败。

关闭调试器，保留 Outpost 防火墙时，删除临时文件，运行软件并拦截测试，我们得到了相似的结果。

由此说明，DNS 查询在 UltraSurf 软件的连接中扮演了承前启后的重要角色。它使得 UltraSurf 能够动态获取 IP，连接方式多样。

### 3)IE 浏览器的修改

在调试中发现，调试到软件不完全运行时关闭调试器，以后使用 IE 浏览器时有时打不开任何网页，有时可以打开，但是完全运行软件后 IE 浏览器就能打开网页，这样的现象促使我们寻找原因。

在 IE 浏览器不能打开任何网页时察看 IE 浏览器的设置，发现在连接->局域网设置->为 LAN 使用代理服务器的选项中代理服务器被设置成 127.0.0.1:9666，在反汇编代码中查找相关字符串，果然找到了这两个字符串。

根据 setInternetOption 等函数的定义和位置，可以判定，UltraSurf 软件在进行尝试连接是否正常以前，将 IE 浏览器的代理服务器设置成了 127.0.0.1:9666，以达到测试利用代理服务器是否能正常连接的目的。在退出软件时，软件再将 IE 浏览器的代理服务器选项取消。

### 4) 客户端发往服务器端的信息初步分析

由上节的分析结果，发现客户端发往服务器端的数据中，末 13 个字节是相同的。跟踪这 13 个字节的生成过程，发现在生成这 13 个字节的生成过程中，前两个字节是固定的常数，而后 11 个字节是通过算法生成的。这 11 个字节的生

成过程中，尽管调用了 rand()函数，却每次都使用一个固定的数作为 srand()的种子，而不是像以前使用随机数时利用当前时间或计算机运行的时间作为种子。

进一步在这个作为种子的数（八位十六进制数）被存放的内存地址设置内存写入断点，这样可以观察到何时这段内存被写入数据。根据这段内存被写入的方式，发现这个数是由一段算法从一个长度为 0x20 的字符串变化而来，而这个字符串在每次运行时是相同的，但在程序载入时该字符串所在内存地址值为 0，所以这个字符串也是在运行过程中生成的。

用内存断点的方法跟踪字符串的生成过程，发现在调用了 CreateFile 和 DeviceIOControl 两个 API 后，返回的数似乎与该字符串的生成有关。

这两个 API 以前没有接触过，查找 MSDN 和相关资料，以及结合动态反汇编调试时调用 API 所传入的参数，得出了这两个函数的作用。在 UltraSurf 软件中，利用 CreateFile,打开机器上的第一块硬盘，返回该设备的句柄。用这个句柄作为参数，调用 DeviceIOControl，指定一块内存区域作为输出，存储调用这个函数所输出的硬盘参数信息。利用这个参数信息通过算法（见 3.5.6 节）生成前述的八位十六进制数。

由于这个生成的数是机器相关的，所以可以用它作为一个标识符，从网络传输的角度来说，就可以唯一地标示这台发送请求的机器，所以一个局域网内的两台机器，即使使用一个公共 IP，也可以同时使用 UltraSurf 软件而被服务器端正确区分。猜测之所以要在发出的密文后附加一段和机器有关的密文，是因为服务器需要判定数据包是否自同一台机器发出，可能有防止攻击的考虑，也可能是传输信息的需要。

### 3.4 UltraSurf 软件的工作原理

UltraSurf 软件是一款典型的穿透类互联网软件，它能穿越目前的网络监控和封锁机制的根本原因是它通过加密连接到代理服务器，利用代理服务器，将本地机器上的网页请求转化为代理服务器上的网页请求，由于代理服务器位于国外，不在被封锁的区域，所以就可以在本地访问一些目前中国大陆通过正常途径不能访问的网页，然后再通过加密传输将代理服务器接收到的网页信息传输到本机。

### 3.4.1 UltraSurf 软件的网页请求

图 3-10 说明了 UltraSurf 软件能够突破现有封锁完成一次网页请求的过程。如果要对其进行封锁，关键就是要弄清怎样获得代理服务器的 IP 地址。

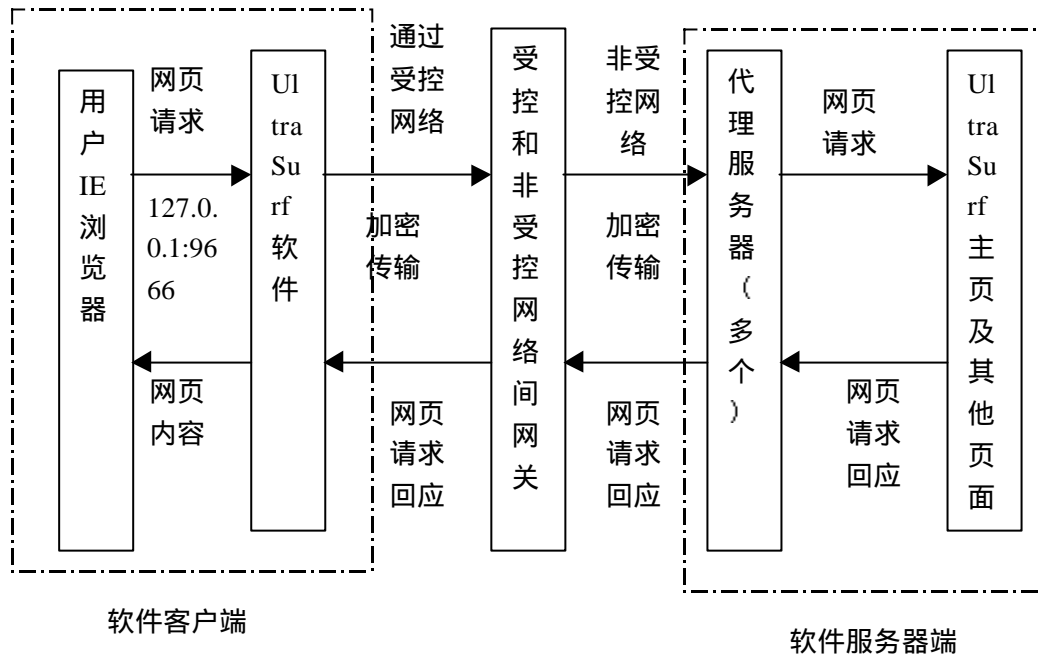


图 3-10 通过 UltraSurf 软件，浏览 Web 页面的过程

Fig. 3-10 Browse web through UltraSurf

这个过程模型是通过对软件客户端的逆向分析而推断出来，对于服务器端的描述不能做到精确，但能够解释 UltraSurf 软件的网页传输穿透现有封锁手段的原理。因为是加密传输，所以传统的关键字过滤对其传输的信息无能为力。

### 3.4.2 UltraSurf 软件获得代理服务器 IP 过程

图 3-10 中的代理服务器地址获得过程是 UltraSurf 软件的关键功能之一。

分析表明，该软件一共存在 4 种获得代理服务器 IP 的方式：

#### 1) 通过读取缓存文件获得代理服务器 IP

其简单流程为：读取计算机临时文件夹中的缓存文件，其路径是通过算法（见 3.5.1 节说明）得出的，读取该文件的内容，对其进行解密，得到上次成功连接所保存的代理服务器 IP，利用这些 IP 进行加密通信。

#### 2) 通过 DNS 查询

其简单流程为：软件运行时，解密软件内置的 DNS 服务器 IP 地址和内置的用于 DNS 请求的域名（例：ns1.flade753d.net），和 DNS 服务器建立连接，软件同时产生多个线程向这些 DNS 服务器发送代理服务器域名请求，等待 DNS 服务器成功返回 IP，然后利用算法对返回的 IP 进行变换，得出代理服务器 IP，最后与新获得的代理服务器 IP 进行加密通信。

### 3) 基于 gdoc 个人空间的代理查询

其简单流程为：通过目前网络上一些门户网站提供的个人服务项目作为载体，存放其代理服务器的信息。我们通过反汇编发现软件内置了 Google doc 的域名为 [https://docs.google.com/View?docid=dd4gbd38\\_6c8fpk2](https://docs.google.com/View?docid=dd4gbd38_6c8fpk2)。如果 DNS 代理查询方式得不到可用的代理服务器信息，软件将使用 google doc 查询方式。首先连接到 <http://docs.google.com>，然后将上述域名信息中的最后一部分个人身份标识信息随机拆分成数段进行发送，全部发送完毕后等待 HTTP 应答。收到应答后，将其中的非字母字符全部去掉，然后以第 1-8 个字符与第 13-20 个字符为参数，进行变换即可得到真实的代理服务器 IP 地址<sup>[8]</sup>。

### 4) 直接与内置加密代理通信

其简单流程为：首先解密软件内置的代理服务器 IP，然后直接与解密的 IP 进行加密通信。与代理服务器通信的过程中也会返回新的代理服务器 IP。软件共内置了 5 个特定 IP：

211.74.78.17  
66.245.217.9  
66.245.217.227  
66.245.196.247  
118.168.50.105

UltraSurf 软件的代理服务器获得过程先执行(1)；失败后执行(2)；再失败则执行(3)；(1),(2),(3)均失败最后执行(4)。

图 3-11 描述了这四种方式的工作过程。

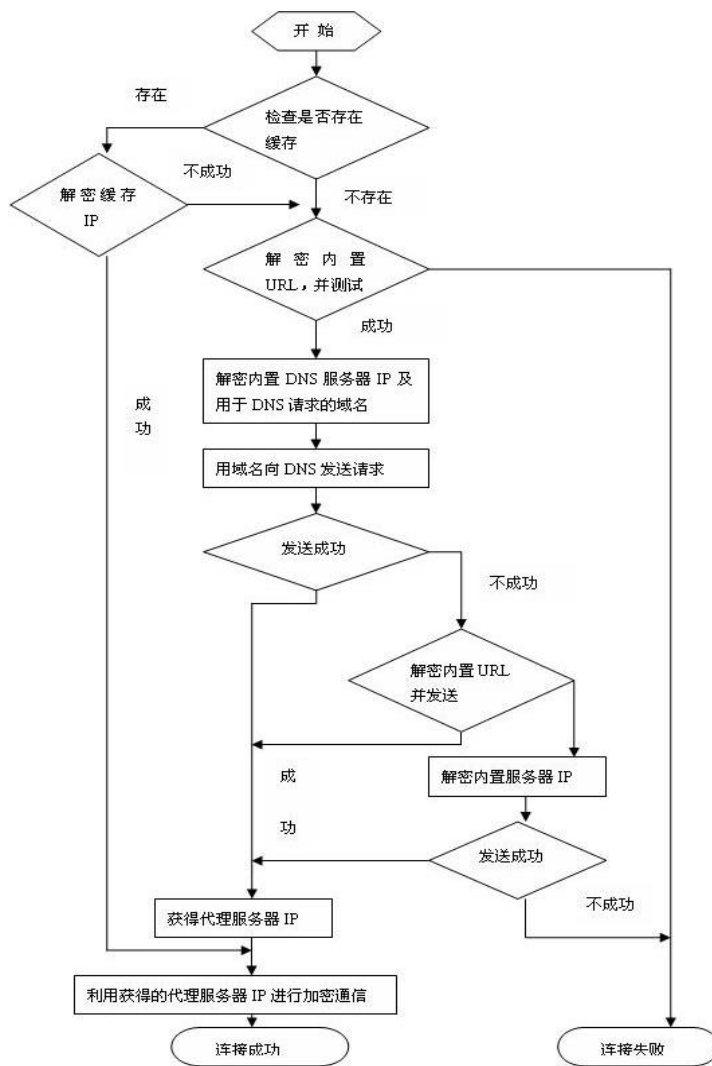


图 3-11 UltraSurf 软件的工作流程

Fig. 3-11 working process of UltraSurf

图 3-11 说明:

软件内置加密的用于 DNS 请求的域名 40 个, DNS 服务器 IP 地址 351 个。软件开始运行时, 首先设置 IE 浏览器代理服务器为 127.0.0.1:9666, 读取上次成功运行时写入的磁盘缓存文件, 若缓存文件存在, 则读取上次使用过的代理服务器 IP, 然后尝试连接, 若能连接上则连接成功。若缓存文件不存在或者连接失败, 则根据磁盘序列号等生成缓存文件的文件名, 建立一个空文件, 准备本次成功连接后写入 IP 地址。解密 DNS 域名请求 (类似: ns1.flade735d.net 的 40 个域名), 然后启动新线程, 测试对外网络连接, 如果连接正常, 则解密内置 DNS 服务器 IP 地址, 向 DNS 服务器发送 DNS 域名请求 (UDP 连接), 得到代理服务器 IP, 经过转换得到真实 IP, 写入内存。如果不能从 DNS 服务器

获得代理服务器 IP 或获得的 IP 不能成功连接, 则解密内置代理服务器 IP, 与其进行连接。成功连接后, 打开 UltraSurf 主页, 将使用过的代理服务器 IP 写入缓存文件以便于下次直接连接, 连接即完成。

图 3-11 标注了三个获得新的代理服务器 IP 的地方, UltraSurf 软件正是通过 DNS 服务器, gdoc 以及代理服务器返回 IP 这三种方式来不断动态获得新的代理服务器 IP 的, 这是它能够突破 IP 封锁方式的原因。这三个返回 IP 的方式均为加密方式, DNS 服务器返回的 IP 虽然通过抓包结果看是一个 IP 形式, 却要经过变换才是实际测试连接的新代理服务器 IP; 与代理服务器的通信则原本就是加密方式。

写入临时文件中的 IP 一般都是动态获得的新 IP, 并且加密。

### 3.4.3 UltraSurf 软件客户端的工作流程

UltraSurf 软件客户端在计算机上第一次运行 (即以前没有成功运行过 UltraSurf 软件) 和后续运行 (以前成功运行过 UltraSurf 软件) 的流程是有区别的。所以, UltraSurf 软件客户端在做某些操作前先检测操作对象是否存在, 例如文件 1, 文件 2, 文件 3 (3.3.6 节) 以及注册表配置信息。

目前分析出的客户端工作流程可以描述如下:

- 1) 软件首先检查临时文件夹内是否存在文件 1, 文件 2, 若有, 则删除这两个文件, 重新生成同样文件名的两个文件, 写入内容。读取注册表内配置信息, 若无配置信息存在, 则写入默认配置信息;
- 2) 解密内置 DNS 域名字符串, 备用;
- 3) 修改 IE 浏览器, 将局域网代理设置为 127.0.0.1:9666;
- 4) 初始化 UltraSurf 界面;
- 5) 利用文件 1, 2 内容生成随机 IP, 生成多个线程, 发送空内容 TCP 报文, 测试此时是否能够通过 127.0.0.1:9666 和这些随机 IP 连接;
- 6) 若测试失败, 显示 “网络连接有问题” 并终止后续操作;
- 7) 测试文件 3 是否存在;
- 8) 若存在, 读取文件 3 存储的上次成功连接的代理服务器 IP 并解密。若不存在, 建立一个空文件 3, 转到 (10);
- 9) 利用多线程与文件 3 存储的 IP 通信, 测试是否能够连接。此连接为 TCP 空包。转到(15);
- 10) 解密 DNS 内置服务器, 多线程 UDP 方式向其发送 DNS 域名请求,

然后接收返回的 IP;

- 11) 将返回的 IP 利用算法转为另一个 IP, 这是真正代理服务器的 IP, 测试这些 IP 是否能够连接;
- 12) 如果这些 IP 能够连接, 转到(15);
- 13) 解密内置 URL, 测试是否能够连接。如果连接成功, 转到(15);
- 14) 解密内置代理服务器 IP, 测试是否能够连接。若不能连接, 显示: “连接服务器失败”, 并终止运行;
- 15) 与测试成功可以连接的代理服务器地址用 443 端口通信, 通信的内容包括网页请求以及获取新的代理服务器 IP;
- 16) 将获得的新的代理服务器 IP 写入临时文件夹的文件 3;
- 17) 用户关闭软件时, 删除临时文件夹的文件 1, 文件 2, 并取消 IE 浏览器的代理服务器设置。

### 3.5 UltraSurf 软件部分算法说明

在分析 UltraSurf 软件时发现了很多算法, 它们大多数与数据的加密解密, 网络传输数据的生成等有关, 大多数利用了随机数并且不是已有的加密算法。

根据 UltraSurf 软件的工作流程, 主要分析了以下对分析有重要作用的算法:

- 1) 临时文件名生成算法
- 2) 数据解密算法
- 3) 内置 DNS 域名解密算法
- 4) 用于加密辅助信息的类 RC4 算法
- 5) 生成发送信息算法一
- 6) 生成发送信息算法二

#### 3.5.1 UltraSurf 软件临时文件名生成算法

UltraSurf 软件临时文件名生成算法的作用是产生一个 8 位临时文件名, 然后在系统的临时文件目录下生成一个以此命名的文件, 存放加密过的数据, 以备下次使用。该算法也用于生成注册表的键值名。该算法有两个参数, 一个是人为设置的十进制数字, 每次数字不同, 将产生不同的文件名, 目前在程序中, 我们发现它只使用了 2, 4, 8, 9 四个数字。生成或读取文件 3 时, 参数为 2,



生成或读取文件 1, 2 时文件参数名分别为 8, 9, 参数为 4 时生成注册表键值, 用于保存 UltraSurf 软件在 IE 浏览器上的设置。

第二个参数是本地磁盘驱动器中 C 盘的卷序列号, 可以通过在命令提示符下运行 vol 命令查看到。

获得这两个参数后, 运行固定的算法:

首先, 对传入的两个参数执行一组固定的算术运算, 它的效果可以等价于一个函数  $F(\text{参数 } A, \text{参数 } B) = C$ ,  $C$  是一个新数值。

其次, 在算法中固定了一个内置密码表, 将新值  $C$  和这个数组中的每一个元素进行另一组算术运算, 它的效果等价于另一个函数  $G(\text{新值 } C, \text{数组元素 } [i]) = \text{新值 } D[i]$ , 该运算将执行 8 次, 即  $i$  从 0 到 7

最后将新值  $D[i]$  顺序拼起来, 得到一个字符串, 就是临时文件名。

算法中使用到的变量定义:

para: 32 位整型, 可变参数

vol: 32 位整型, 硬盘卷序列号

num: 密码表数组, 元素类型 char, 长度为 6, 程序内置

file\_name: 临时文件名数组, 元素类型 char, 长度为 8

算法流程的描述如下:

- 1) 为 para 赋值 (2, 4, 8, 9 其中一个数)
- 2) 为 vol 赋值, 调用系统 API 读取硬盘 C 卷序列号
- 3) 令  $\text{vol} = \text{vol} \wedge (\text{para} * 32)$
- 4) 令  $\text{vol} = \text{vol} \wedge 0x801$
- 5) 令  $\text{vol} = \text{vol} + \text{para}$
- 6) 若 vol 为奇数, 临时文件名数组第一个元素  $\text{file\_name}[0] = 0x7E$ , 否则为  $\text{vol} / 2$  的值对十进制 26 求余, 再加上  $0x41$  (变成一个大写字母)
- 7) 临时文件名数组第二位到第七位根据内置数组 num 中元素和 vol 进行运算得到, 计算方法为: 临时文件名第  $i$  位字符等于 vol 对 内置数组 num 的第  $[i - 1]$  位求余, 再对十进制 26 求余, 再加上  $0x61$  (变成一个小写字母)
- 8) 临时文件名的第八位 (最后一位), 由 vol 对内置数组的第二位求余, 再对十进制 26 求余, 再加上  $0x61$  (变成一个小写字母)
- 9) 算法结束, 生成了临时文件名

### 3.5.2 UltraSurf 软件解密算法

本算法用于 UltraSurf 软件内置数据（包括内置 DNS 服务器 IP 地址和代理服务器 IP 地址）以及文件 3 的解密。

根据分析，UltraSurf 的加密数据具有以下特征，它是由长度具有一定限制的加密数据加上 8 字节密码表组成，8 字节的密码表附在加密数据之后。解密的时候，算法需要用到这 8 字节的密码表配合算法进行解密。

解密算法有三个参数，第一个参数为加密数据区数据长度，第二个参数为密码表的前四个字节，第三个参数为密码表的后四个字节。算法得到参数后，每次取加密数据的一个字节去和密码表的某一字节作算术运算，即可以用一个函数描述为  $F(\text{加密数据}[i], \text{密码表}[j]) = t$ ，得到新的字节  $t$  之后，用  $t$  再去和密码表作算术运算，更新密码表，最后，用变化后的密码表和原来的加密数据按字节逐个异或，即可得到解密后的明文数据。

密码表是附在整个加密数据的最后部分。然而作为解密函数参数的两个值，并非附在密文之后的 8 个字节，程序在使用前还要对这 8 个字节做一些处理。

算法具体流程如下：

- 1) 读取加密数据
- 2) 读取加密数据对应的密码表
- 3) 将加密数据的长度和 0xFABEBABE 异或
- 4) 将密码表的后四个字节和第 3 步的结果异或，得到 8 个字节作为解密函数的密码表参数二和参数三
- 5) 将密码表的前四个字节和 0x3F6CB254 异或，后四个字节和 0xAE985D36 异或
- 6) 判断密码表此时是否全部为 0，若为 0，则使用内置密码表：前四字节的 0x78B4FEAE，后四字节的 0x3DCF578A
- 7) 进行一个次数为加密数据长度的循环，循环算法为：
  - 7.1) 初始化 index 变量为 0
  - 7.2) 用 index 索引查密码表，得到密码值
  - 7.3) 密码值与循环计数变量 counter 作算术运算，公式为：
$$\text{table\_}[\text{index}] \wedge = (\text{counter} / 16) | ((\text{counter} / 16) * 16) ;$$
  - 7.4) 通过循环计数变量 counter 获得密文的一个字符，还原密文，还原算法为

```
plain_[counter] = table_[index] ^ cipher_[counter];
```

7.5) 更新 index 的值, 算法为:

```
index = (cipher_[counter] % 7) ^ index;
```

7.6) 判断是否解密完毕, 若是, 结束, 写回所有的明文, 若否, 则回到 7.2)

8) 循环运行完毕, 解密即完成

### 3.5.3 UltraSurf 软件版内置 DNS 域名解密算法

自一段自运行初始即存在密文内容的内存地址开始, (软件内置) 将每 32 位数值与 0x00 比较。在此 32 位不为 0 前, 将此 32 位的数值与 1F 异或, 即得到实际域名的 ASCII 数值。

### 3.5.4 UltraSurf 软件辅助信息加密算法

分析发现, 在软件运行的过程中, 对一些辅助记录软件动作的信息, 如“2008-05-05 18:40:00,send UDP query to 58.9.3.4”, 软件都将它们记录在内存中。可能是为保密的考虑, 使用了一个算法对这些辅助运行信息进行加密。

这个算法实际效果尚不明显, 但可以作为软件逆向分析中的辅助, 若能读取内存还原出信息可以为调试工作带来很多方便。

这个算法基本是 RC4 算法的变体。

RC4 算法分初始化算法和伪随机子密码生成算法两部分。

算法初始化部分打乱一个 256 字节的初始密码表, 随机将一个顺序位置的字节和一个随机生成位置的字节交换。

密码生成算法则利用这个密码表, 经过与密钥的处理后得到子密码, 然后和明文异或而得到密文, 解密过程完全相同。

在 UltraSurf 软件中, 这个密码表初始为通过一个简单函数生成的 00-FF 的连续 256 个字节, 经过算法初始化部分打乱的密码表还可以作为下一次调用这个算法时使用的密码表。这样保证了密码表的随机性。

利用这个密码表, 算法设置一个指针在密码表的起始地址, 将这个指针位置的字节与一个随机生成的字节交换, 递增指针, 然后利用这两个字节, 通过一系列运算生成一个子密钥, 将软件运行信息字符串加密, 并按顺序保存在某一固定位置。

### 3.5.5 UltraSurf 软件发送到代理服务器信息生成算法一

前面介绍, UltraSurf 软件利用 443 端口发送到代理服务器的数据包中, 共有 14 个字节利用同一个算法, 先后分 8 字节和 6 字节写入。下面的算法描述了这些数据的产生过程。

算法使用一个要写入数据的长度的常数为参数, 目前只发现 8 和 6 两个值, 以及一个密码表基址。算法还通过压入堆栈的方式保存要写入数据的目标地址, 查询密码表的地址 (下称地址 1), 可变密码表的地址 2 个 (下称地址 2, 地址 3, 因为它们作用相同, 所以不具体区分), 地址的生成是随机的 (通过 malloc 动态分配内存)。每次循环后在堆栈保存 1 个上次使用过的字节供下一次循环使用。

算法的主要过程是取地址 2, 地址 3 的各 1 字节数据, 将数据相加, 利用相加后的数据作为查询密码表的偏移量, 加上密码表的基址为地址 1, 然后取写入地址中的原始数据, 与地址 1 中的数据异或, 得到最终发送出去的数据 (1 字节), 循环长度次, 一次写入完成。

算法的具体流程如下:

- 1) 检查数据长度是否为负值。
- 2) 若为负值, 则减去 1, 再取反, 取最后 1 字节值。
- 3) 取当前计数加上密码表基址处的地址为地址 2。
- 4) 取地址 2 处的字节, 加上上一次循环的字节数据, 得到一个新数据, 保存入堆栈准备下次循环使用。
- 5) 此数据加上密码表基址得到地址 3。
- 6) 交换地址 2 和地址 3 处的字节数据, 达到修改密码表的功能。
- 7) 将交换的两字节相加, 取后 1 字节, 检查是否为负值。
- 8) 如果为负值, 则将其减去 1, 取反, 取最后 1 字节, 再加 1
- 9) 用 7) 的结果加上密码表地址, 取该处字节。
- 10) 取目标写入地址加上计数器的值的地址值处的字节。
- 11) 将 9)、10) 得到的字节异或, 结果写入目标地址处。
- 12) 判定计数器值是否到达目标长度 (8 或 6), 若未到则继续循环。

### 3.5.6 UltraSurf 软件发送到代理服务器信息生成算法二

在 UltraSurf 软件 生成发送到代理服务器的数据时, 有 13 个字节是相同的。

其中, 前两字节是固定的常数 08, 01, 猜测用于标示一段信息的开始。后面 11 个字节是用一个算法从一个固定字符串生成的。此算法虽然利用了 rand 函数, 但是利用了一个固定的, 从机器信息推断出的种子, 生成的信息对于同一台机器都是相同的。

下面是对此算法的描述。

- 1) 调用 CreateFile 函数, 获得机器第一块硬盘的句柄。
- 2) 调用 DeviceIOControl 函数, 传递获得的句柄和 SMART\_RCV\_DRIVE\_DATA 参数, 将获取的硬盘参数写入某块内存。
- 3) 调用系统函数 netbios 在内存中写入 10 个字节, 和网卡 mac 地址有关。
- 4) 参数为循环次数, 为 0A+ 字符串 "WD-WMAM9DZ12046" 的字节数, 共 0x1E 次。

指向字符串 (前 10 个字节由 3) 生成, 后 20 个字节为 "WD-WMAM9DZ12046") 的指针, 常数 0xF8C9。变量 EAX (初始化为 0, 存放结果), ECX (初始化为 0xF8C9), ESI (初始化为 0, 计数器) EDI (存放内存读取数据)

- 5) 依次取字符串一个字节保存入 EDI,  $EAX * ECX + EDI$ ,  $ECX = ECX * 0x5C6B7$ , 修改计数器, 继续循环。

- 6) 最后生成 8 字节的 EAX, 写入内存的固定位置。由于磁盘和网卡参数是固定的, 函数中的参数是固定的, 所以生成的 EAX 是一个恒定的数。

在以后生成发送信息的后 11 字节时利用了这里的 EAX 作为种子。

这里的发送信息包含了机器信息, 也就是“一机一码”, 可以用来标识不同的使用 UltraSurf 软件的机器, 可以保证在一个局域网内的不同机器正常使用 UltraSurf 软件。

### 3.6 软件分析成果

本章对 UltraSurf 软件进行了分析, 得出了该软件的许多关键信息, 为控制该软件在互联网上的泛滥积累了重要的知识和经验。下面是一些主要的分析成果:

#### (1) 软件编写语言

掌握软件的编写语言是识别软件内部结构的关键, 经过分析得到, UltraSurf 软件的编写语言是 Visual C++ 6.0。

## (2)软件加壳

目前的软件普遍使用了加壳技术进行保护，经过我们分析，UltraSurf 软件使用了 UPX 加壳技术进行保护，同时我们也对 UltraSurf 软件进行了脱壳处理。

## (3)软件关键结构

经过反汇编分析，UltraSurf 软件是基于 MFC 的 Windows 应用程序，其中采用了多线程技术。

## (4)软件网络通讯技术

经过反汇编分析和网络监控发现，UltraSurf 软件使用了 Winsock2 API 函数进行网络通讯，查询技术使用的是 TCP 和 UDP 两种方式，并未使用自有通讯协议。

## (5)软件内置 DNS 和 IP

经过反汇编分析，得到了软件内置的所有 DNS 域名及进行 DNS 查询的 IP 地址。

## (6)软件内置的加密代理 IP 及 URL

经过反汇编分析，获得了软件内置的加密代理 IP 及 URL，这些方式是在 DNS 查询失败后直接进行尝试连接的备用加密代理地址。

## (7)软件启动和通信的过程

经过分析，得出了 UltraSurf 软件启动并建立通信的整个过程。

## (8)软件算法

通过分析，得出了软件所使用的加解密算法。

## 第四章 封堵方案设计与验证

### 4.1 穿透类软件的特点与分析方法

通过前面对 UltraSurf 软件进行分析，我们可以发现它是通过网络查询来动态获得加密代理 IP 地址，然后使用加密代理技术进行安全的通信。

#### 4.1.1 加密代理技术

分析表明，这类软件提供信息传输服务的核心技术依然是传统的代理服务技术，然而它们之所以能够逃脱封锁，是因为使用了隐蔽的连接技术和传输加密技术。图 4-1 显示了该类软件所使用的加密代理技术的整个工作过程。

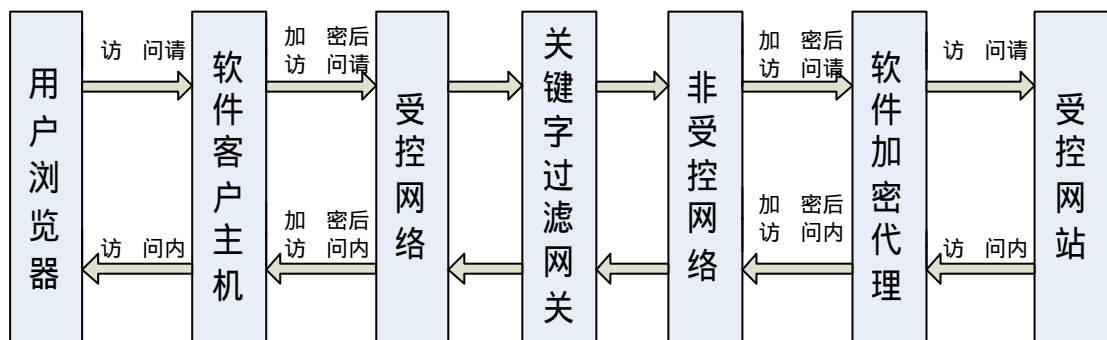


图 4-1 加密代理技术示意图

Fig. 4-1 The Flow Chart of Encrypting-Agent Technology

#### 4.1.2 穿透类软件的分析方法

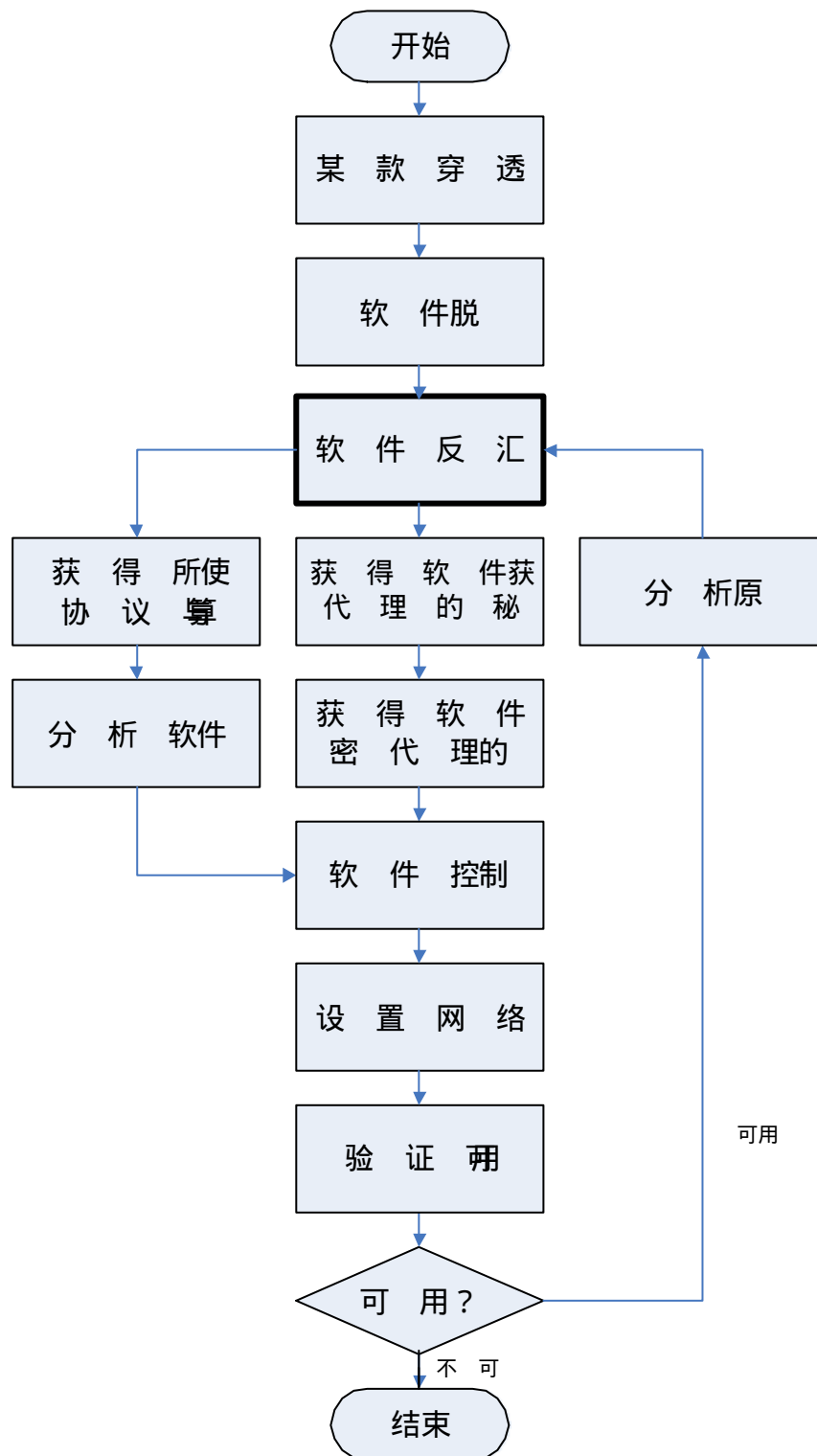


图 4-2 软件分析流程图

Fig. 4-2 The Flow Chart of Software Analysis



针对这类软件的特点，我们设计了对它们进行分析的通用流程和方法。如图 4-2 所示，首先根据现实的需求选定需要分析的软件；然后对该软件进行反汇编分析，得出其获取加密代理的机制和秘密信息，并且获得其所使用的加密协议与算法，分析其可以利用的漏洞（如果存在的话）；下一步是针对分析得来的 IP 列表和可利用漏洞，进行封堵方案的设计，并在实验室的网络环境下进行试验：先对防火墙进行设置，再检验软件的可用性，如果可用，则需要重新进行反汇编分析、重新设计封堵方案，如果不可用，则表明封堵方案有效。

## 4.2 常用封堵方法

目前，经常采用的封堵方案有 IP 地址封锁、关键字过滤和域名劫持<sup>[16]</sup>等。这几种方法各有其特点。

### 1) 网关 IP 封锁

网关 IP 封锁是从 90 年代初期就开始使用的一种方法，国家会将宣扬反动言论、危害国家安全的站点进行 IP 封锁。这些站点大多位于国外，对于国外的站点，只要在国家级网关出口处进行封锁就可以了。

对于 IP 封锁，用普通代理技术就可以绕过。方法是找到一个 IP 不被封锁的、位于设置了 IP 封锁的网关之外的代理，通过这个代理就可以访问被封锁的 IP 站点的信息了。

### 2) 主干路由器关键字过滤

2002 年以后，随着关键字过滤的成熟，出现了功能强大的关键字过滤路由器。各国均不同程度的开始采用这类路由器来净化互联网环境。中国采用了这样一套系统，通过各因特网服务提供商来具体实施。这套关键字过滤系统能够从计算机网络系统中的关键点（如国家级网关）收集分析信息，过滤、嗅探特定的关键字，并进行智能识别，检查网络中是否有违反安全策略的行为。利用这些设备主要进行网址的过滤和网页内容的过滤，如果符合设定的规则，则向用户发送 ACK-FIN，自动打断用户与服务器的会话连接，使数据流中断，而在终端电脑上会显示主机无法识别。不同的 IDS 甚至有可能在一段预定或随机的时间内试图阻止从用户主机发出的所有通信<sup>[18]</sup>。

所以在访问不法网站或者数据流里含有敏感字符时，会被提示“该页无法显示”，随后在 5-15 分钟的时间内无法用同一 IP 浏览此域名或该 IP 地址上的内容，屏蔽时间据猜测和敏感词等级以及所属网站有关。此种过滤是双向的，

也就是说, 网关内含有关键词的网站在网关外不可访问, 网关外含有关键词的网站在网关内不可访问。

关键字过滤的弱点就是对已加密的信息无能为力, 而网址的关键字和网页的关键字都可以用不同的手段来加密, 从而使这样的信息过滤系统从根本上失去作用。

### 3) 域名劫持

域名劫持是对互联网不良信息进行封锁的另一种常用手段, 即在自己的网络范围内把被访问的域名(网络地址)改成假的 IP 地址, 让自己网络范围的用户均无法正确访问所请求的地址信息。域名劫持的主要弱点是它不很稳定, 在某些高速网络环境下, 返回真实的 IP 地址比劫持软件提供的假地址要快。而且, 在网上查询域名和正确的 IP 非常容易, 利用国外的在线 IP 地址查询服务, 可以容易地查到网站的真实 IP 地址。可以采用域名劫持的前提是用户要访问的域名是可见的, 对加密后的域名则毫无办法。

## 4.3 封堵方案的设计

对于 UltraSurf 软件, 通过反汇编分析我们知道, 第一次使用时, 该软件首先读取保存在程序内部的 DNS 服务器 IP 和域名, 然后将一个域名发送给一个 DNS 服务器 IP 进行查询, DNS 返回含有 IP 的数据, 经过 UltraSurf 软件变换出加密代理 IP, 此时软件将加密数据发送给加密代理服务器, 而该服务器并不具备加密信息的处理能力, 但是它提供了将信息转发给极景公司的中转功能。一般来说, 加密代理服务器位于网关外, 虽然我们能封锁从网关内到极景公司的路径, 但对于加密代理服务器到极景公司服务器的路径却无能为力, 这样 UltraSurf 软件就可以逃避封锁。那么, 我们可以考虑切断 UltraSurf 软件向外发送信息的两条路径: DNS 查询和加密代理连接。这两条路径中, 加密代理是动态变化的, DNS 查询是静态的, 显然, 对 DNS 进行封堵更为容易, 我们可以直接封锁所有软件内置的 DNS 服务器 IP 地址。

UltraSurf 的封堵方案如下:

- 1) 利用 IP 过滤技术, 封锁所有 DNS 服务器的 IP 地址;
- 2) 利用 IP 过滤技术, 封锁所有已知的加密代理 IP 地址;
- 3) 利用关键字过滤技术, 封锁所有已知 URL。

## 4.4 封堵方案的部署与验证

我们在实验室内对 4.3 节所提出的封堵方案进行了部署，并且验证了该方法的有效性与可行性。该实验所使用的网络环境拓扑图如图 4-3 所示。

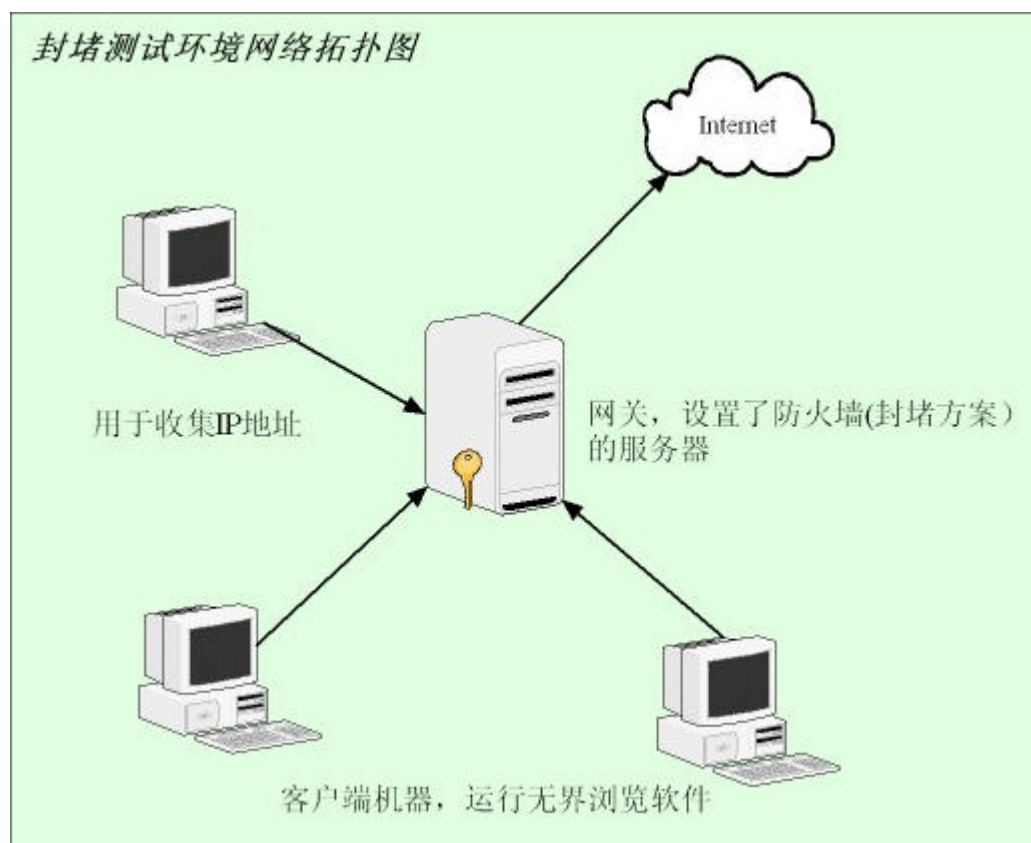


图 4-3 网络拓扑图

Fig. 4-3 Network Topology

封堵设备目前是基于 PC 的，利用一台运行服务器程序（Fedora 6 Linux 操作系统）的主机作为网关进行。由于 Linux 操作系统强大的网络功能，实际上是利用这个网关作为防火墙，利用 Linux 的 iptables 软件添加网络访问规则以达到封锁分析出的 IP 地址和 URL 的功能。

封堵的实验环境包括在一个局域网中的机器，利用前述运行 Linux 操作系统的主机作为网关，机器通过网关与外部网络连接。主机上运行封堵服务器程序，可以实现自动添加 IP 地址，判断 IP 地址区域等功能。

因为 UltraSurf 软件具有动态从连接中获取 IP 的特性，所以在正常连接上 UltraSurf 软件时，可能返回不在封锁列表中的代理服务器 IP 地址。

对于此类 IP 地址的收集，暂时采用了一个改进方法——缓存机制：利用某一台能够连接上 UltraSurf 网站的机器，编写程序，反复、自动运行该软件来收集缓存中的 IP 地址，并与封堵服务器通信，发送收集到的 IP 给服务器，服务器动态添加 iptables 规则，即对封堵 IP 列表进行动态更新，这种方法可以尽可能地收集较多的代理服务器地址，使封堵更加成功。

#### 4.4.1 封堵测试流程

分两种情况进行测试。

测试顺序如下：

##### 1 第一次使用：

在两台没有使用过 UltraSurf 的配置和网络条件均相同的机器上使用该软件，分别在网关上加载封堵程序和不加载封堵程序，将两次测试情况对比。若加载封堵程序后机器无法连接，在网关上关闭封堵程序，再次测试。

（最好在一次测试完毕之后，两台机器对换，重复上述测试）

##### 2 存在缓存后再次使用：

（注：由于条件所限，本次测试中的缓存读取并提交 IP 到服务器的程序部署在与测试机相同的机器上，更大规模的测试方案应该使用另一台机器运行该程序）

在使用过 UltraSurf 并成功连接过网络的机器上进行测试，在网关上加载封堵程序，首先不开启缓存读取机制，测试是否能够连通，若能连通，开启缓存读取机制，动态添加 IP，再次进行测试。若加载了缓存读取机制后不能连接，此时去掉缓存读取机制，继续测试，若此时还不能连接，去掉封堵方案，测试是否能够连接。（在进行本次测试时最好使用另外一台机器访问待测试网站，确保它们能够在没有过滤的情况下连接成功）

测试中使用的正常网站包括：

www.yahoo.com          cn.profiles.yahoo.com          www.msn.com  
spaces.msn.com          spaces.live.com          flickr.com

非法网站包括：

www.qxbbs.org（在此网页上进一步选择各个穿透类软件主页的连接）  
www.dajiyuan.com

#### 4.4.2 封堵结果

根据测试结果表明，在实验的网络环境下，UltraSurf 软件能够很好地被屏蔽，表 4-1 和 4-2 表明了多次测试后的结果。

表 4-1 第一次使用 UltraSurf 的测试结果记录

	网关加载封堵程序		网关不加载封堵程序	
	正常网站访问	非法网站访问	正常网站访问	非法网站访问
第一次连接	完全正常	无法访问	完全正常	除少数情况无法访问外，能够正常访问
撤去封堵程序后连接	完全正常	完全正常	/	/

表 4-2 存在缓存后再次使用 UltraSurf 的测试结果记录

测试顺序	正常网站访问	非法网站访问
不开启缓存读取机制	完全正常	大部分情况下无法访问，小部分情况可以访问
开启缓存读取机制	完全正常	无法访问
去掉缓存读取机制	完全正常	无法访问
去掉封堵程序	完全正常	完全正常

结果说明目前的封堵方案在局域网环境内封堵效果良好。少数情况下非法网站访问可能受其他因素的干扰，如软件不能正确连接到代理服务器或连接超时等。

#### 4.4.3 代理服务器 IP 地址分布

通过对 UltraSurf 软件代理服务器 IP 的收集，我们得到了 6000 多个 IP 地址。通过对这些 IP 其所在地域进行查询及统计分析，发现这些 IP 地址主要来至台湾地区和美国。表 4-3 为详细信息：

表 4-3 各地区 IP 地址分布表

台湾	美国	加拿大	德国	法国	澳大利亚	马来西亚
4512	1360	442	103	22	12	6

## 第五章 结束语

随着网络技术和互联网的普及，网络已逐渐成为信息发布和传播的重要基础设施。互联网对于维护国家稳定和安全的作用也越来越被各国政府所重视，因此网络监管部门采取了多种措施来保障互联网上传播内容的安全和健康。然而，近年来出现了一些采用加密技术的突破网络封锁的软件，通过这些软件，任何信息和言论都可以在网络上进行传播，这引起了越来越多网络管理部门的关注。

以 UltraSurf 为代表的是一类典型的“穿透类”软件，这类软件采用了数据加密传输、内置信息加密、动态更新加密代理 IP 等技术，使得现有的网络监管技术难以有效的对其进行控制。这类软件下载后无需安装，直接就可以运行，而且操作简单，正在被越来越多的网民所使用。如果这类软件得到了大规模的使用，并用以传输误导民众、危害国家的言论，将会给国家带来很大威胁。本文的目的就是分析这类软件的典型代表，总结它们的共性，提出封堵这类软件的方法。

本文分析了 UltraSurf 软件，给出了它们启动时的整个流程，并对它们所使用的获取加密代理的技术和加解密技术进行了分析，然后总结了这类软件所采用的通用方法和技术。最后根据分析结果，设计了适合现有网络的封堵方案，并在实验室内对该方案的有效性进行了验证。

在现有工作的基础上，未来我们将在以下几个方面展开进一步的研究：

- (1) 深入研究这类软件通信时所使用的加密算法与协议，找出可以利用的漏洞；
- (2) 设计软件，自动实现用户缓存中加密代理 IP 地址的统计，并自动升级封堵网关上的规则；
- (3) 对该类软件中的其他软件进行分析。

## 参考文献

- [1] 段钢, “加密与解密”, 第二版, 北京, 电子工业出版社, 2003。
- [2] 飞天诚信, “软件加密原理与应用”, 北京, 电子工业出版社, 2004。
- [3] 看雪学院, “软件加密技术内幕”, 北京, 电子工业出版社, 2006。
- [4] 陈昊鹏, “软件逆向工程技术研究”, [博士学位论文], 西北工业大学, 2001。
- [5] E Chikofsky, J Cross, “Reverse engineering and design recovery”, A taxonomy IEEE Software, 1990, 7(1)
- [6] Hassan, A.E., Holt, R.C. “The small world of software reverse engineering”, Reverse Engineering, 2004.Proceedings. 11th Working Conference on 8-12, 2004.11.
- [7] Rainer Koschke. “Software Visualization for Reverse Engineering”, Lecture Notes in Computer Science. Volume 2269, 2002.
- [8] Andritsos, P., Miller, R.J. “Reverse engineering meets data analysis”, Program Comprehension, 2001. IWPC 2001. Proceedings. 9th International Workshop on 12-13, 2001.05.
- [9] Moise, D.L., Wong, K., Sun, D. “Integrating a reverse engineering tool with Microsoft Visual Studio .NET” Software Maintenance and Reengineering, CSMR 2004. Proceedings, 2004.
- [10] Kris Kaspersky, “Hacker disassembling uncovered”, 北京, 电子工业出版社, 2004。
- [11] Kip R. Irvine, “Assembly language for intel-based computers”, 北京, 电子工业出版社, 2004。
- [12] 罗云彬, “Windows 环境下 32 位汇编语言程序设计”, 北京, 电子工业出版社, 2006。
- [13] 郭栋, 孙锋, 唐植明, “加密与解密实战攻略”, 北京, 清华大学出版社, 2003。
- [14] 郭颖, 钱渊, “逆向工程的应用研究和发展”, 信息与电子工程, Vol.2 No.2, 2004.6。
- [15] 唐晓君, 单宝卫, 路莹等, “软件再生工程理论及其新的实现方法”. 大连: 《大连轻工业学院学报》, 第二十卷第四期, 2001.
- [16] 陈恺, 吴国新, “VPN 中隧道交换技术的研究”, 大连, 大连轻工业学院学报, 2003.5。

- [17] 周立萍,孙青岩,陈平,“逆向工程分析技术研究”, 微机发展, Vol.14 No.4, 2004.4。
- [18] 张玉清,“计算机通信网安全协议的分析研究”, [博士学位论文], 西安电子科技大学, 2000.1。
- [19] 李必信, 郑国梁, 李宣东等,“软件理解研究与进展”, 计算机研究与发展, Vol.36 No.8, 1999.8。
- [20] 季军杰,“反汇编环境下函数及其参数、变量的识别技术”, 惠州大学学报, Vol.20 No.4, 2000.12。
- [21] 杨季文,“80X86 汇编语言程序设计教程”, 北京, 清华大学出版社, 1999。
- [22] 张志猛, 庄越挺, 潘云鹤,“面向对象软件的逆向工程”, 计算机研究与发展, 第 40 卷第 7 期, 2003.7。



## 缩略语

PE: Portable Executable 可移植的执行体  
COFF: Common Object File Format 通用对象文件格式  
RVA: Relative Virtual Address 相对虚拟地址  
IAT: Import Address Table 导入地址表  
OEP: Original Entry Point 入口点  
IDS: Intrusion Detection System 入侵检测系统

## 致 谢

在本硕士论文即将结束的时候，我要感谢所有关心、帮助我的老师、同学和亲人。没有你们的帮助，我便不能顺利地完成本文。

首先要感谢的人是我的导师谷大武教授，感谢他两年多来对我学习和生活的无私关怀与悉心指导！谷老师严谨求实的作风、精益求精的态度、宽厚为人的性格给我留下了深刻的印象。在谷老师的关怀、指导与鼓励下，我得以顺利完成本论文的撰写工作。两年多的时间转眼而过，然而谷老师的谆谆教导和伟大人格将使我受益匪浅、永生难忘。在这里，我衷心地祝愿谷老师工作顺利、身体健康！

衷心感谢实验室的陆海宁老师、刘军荣老师，他们严谨认真的学术态度，勤奋负责的工作精神，令人难忘。

感谢丁宁师兄、李炜师姐、任艳丽师姐、张媛媛师姐、顾海华师兄、赵建杰师兄、申坤师兄、罗德祥师兄以及陈帆、段冰、张群、宋宁楠、陈维烨、曾梦岐、李卷孺、尚涛等同学在科研中给予的合作与帮助，正是因为有了他们的帮助，我才能够顺利地完成本文。

感谢学院洪敏老师、朱春玲老师等领导和老师在工作、学习和生活上给予的指导与帮助。感谢班级同学的关心与支持，这使我度过了两年半充实而快乐的时光。正是因为你们的帮助，我才得以顺利地完成了学业，这一切都将会成为我生命中美好的回忆。

## 攻读学位期间发表的学术论文及参与的科研项目

- [1].张磊, 谷大武, 陆海宁, 陈帆 “UltraSurf软件的运行机制分析” ;《通信技术》, 2008 年 9 月录用。
- [2].上海市信息化发展专项资金项目: “穿透类软件控制技术研制”。

# UltraSurf软件的逆向分析技术研究

作者: [张磊](#)  
学位授予单位: [上海交通大学](#)

本文链接: [http://d.g.wanfangdata.com.cn/Thesis\\_D067838.aspx](http://d.g.wanfangdata.com.cn/Thesis_D067838.aspx)