面向软件资源共享的 RDP Server 研究与实现①

Research and Implementation of RDP Server for Software Resource Sharing

沈士根 （嘉兴学院 信息工程学院 浙江嘉兴 314001）

摘要：软件资源共享是保障网络化制造的重要技术。采用中间人攻击的 ARP 欺骗方法，获得了未公开的主要 RDP 数据包。根据 RDP 数据包开发了基于 Linux 的 RDP 服务器 RDP Server（RServer），给出了 RServer 的系统结构，实现了网络化制造的软件资源共享。

关键词：软件资源共享 网络化制造 RDP

1 引言

在网络制造化领域，一些大型国有企业经过长期积累以及国家的持续投入，拥有了大量贵重软件资源，但这些资源并未得到充分利用，而许多中小企业因为资金原因买不起大型贵重软件，这种矛盾限制了区域网络化制造的发展。如能在网络化制造集成系统上构建软件资源共享平台，就可使缺乏软件资源的企业，通过 Internet 利用其他企业的软件进行产品设计等。这样，既提高了软件资源拥有方的软件利用率和企业投资效益，又避免了缺乏软件的企业的重复投资，达到区域资源优化配置目的。针对这种现状，本文提出了一种基于 Linux 的 RDP Server（RServer），通过 RServer 使公网用户可访问内网终端服务器上的软件资源，从而实现资源共享。因为 RDP 协议的不公开性，本文首先利用 ARP 欺骗方法获得了主要的 RDP 数据包，然后根据 RDP 数据包并在参考 Rdesktop[1] 的基础上，设计开发了 RServer。

2 利用中间人攻击 Windows 终端服务

Erik Forsberg 在文献[2]说明了利用中间人攻击 Windows 终端服务的方法，为此开发了基于 Linux 平台并开放源码的 rdpproxy[3]。但 rdpproxy 只实现了 RDP 数据包截获和数据包转发功能，存在不能保证截获 RDP 数据包和得到的 RDP 数据包不易理解的问题。为此，我们在 rdpproxy 的基础上增加了 ARP 欺骗模块和 RDP 数据包解析输出模块。如图 1 所示，把修改后的 rdpproxy 称为 RDP－MITM，内含的 ARP 欺骗模块负责客户端和服务器的 ARP 欺骗，内含的 RDP 数据包解析输出模块对截获的 RDP 数据包进行解析，并以一种易理解的格式输出。

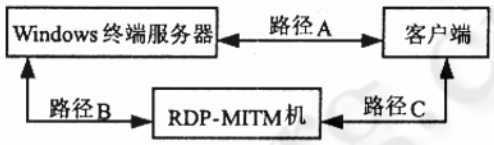

图 1 攻击环境图

在图 1 中，Windows 终端服务器采用 Windows 2000 Server 并安装应用程序模式的终端服务；RDP－MITM 采用 Fedora 4 操作系统；客户端采用 Windows 2000 Professional 操作系统并安装远程桌面连接。攻击流程包括：①RDP－MITM 机以命令行方式启动 RDP－MITM，运行 ARP 欺骗模块，使服务器认为客户端 IP 对应的 MAC 地址是 RDP－MITM 机的 MAC 地址，客户端认为服务器 IP 对应的 MAC 地址是 RDP－MITM 机的 MAC 地址。该模块间隔 30 秒重复执行一次；②RDP－MITM 中的 RDP 数据包截获模块监听 TCP 3389 端口；③当客户端试图连接服务器时，实际连接的是 RDP－MITM 机，然后 RDP－MITM 机运行数据包转发模块转发客户端请求到服务器；④服务器产生一对 $\{PK_1, SK_1\}$ 和随机数 S_random，向 RDP－MITM 机发送 $PK_1 \| S\_$

① 基金项目：浙江省教育厅科研项目资助（编号：20061608）

random；⑤RDP－MITM 产生一对{PK_2，SK_2}，向客户端发送 PK_2‖S_random；⑥客户端产生随机数 C_random，向 RDP－MITM 发送 E_{PK_2}[C_random]；⑦RDP－MITM 机通过 D_{SK_2}[E_{PK_2}[C_random]]得到 C_random，根据 S_random 和 C_random 生成会话密钥。至此，RDP－MITM 得到的会话密钥即是服务器和客户端通信的会话密钥，可解密服务器和客户端之间的所有 RDP 数据包。

3　RDP 数据包

在参考 RFC905[4]、RFC2126[5]、T125[6]和 T128[7]协议基础上，利用上述攻击方法，获得了从建立连接到数据传输的主要 RDP 数据包，如图 2 所示。

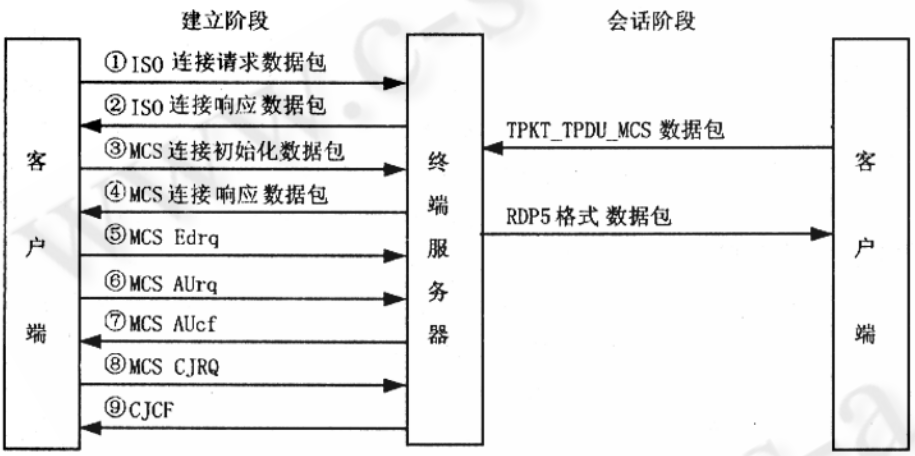

图 2　RDP 数据包

3.1　建立阶段

3.1.1　ISO DP 8073 连接建立数据包

在初始化 RDP 连接时，客户端发出 ISO 连接请求数据包，该数据包是 ISO DP 8073 数据包的一部分，包括：1 字节整型 TPKT version；1 字节整型 TPKT reserved；2 字节整型 TPKT length；7 字节 TPDU；22 字节字符型 mstshash。其中：mstshash 不是 RFC 2126 或 RFC 905 标准的一部分，而是微软给出的 RDP 扩展，主要用于通过 IE 浏览器的 Active X 控件访问终端服务时提供客户端的版本号以及与 Web 服务器的联系。

服务器收到客户端 ISO 请求后，返回一个连接响应，数据包格式也是 ISO DP 8073 的一部分。包括：1 字节整型 TPKT version；1 字节整型 TPKT reserved；2 字节整型 TPKT length；7 字节 TPDU。

3.1.2　MCS 连接初始化数据包

微软在参考 T125 的基础上，对 T125 中定义的 MCS 中的 userdate 中加入了 RDP 客户端的信息。数据包包括：1 字节整型 Calling Domain 值；1 字节整型 called Domain 值；1 字节整型 Upward Flag 值；19 字节 Target Domain Parameters；19 字节 Min Domain Parameters；19 字节 Max Domain Parameters；2 字节整型客户端信息标识；2 字节整型客户端信息长度；2 字节整型客户端 RDP 版本（0X0001 表示 RDP Version 4，0X0004 表示 RDP Version 5）；2 字节整型客户端纵向分辨率；2 字节整型客户端横向分辨率；4 字节整型客户端键盘布局；2 字节字符型客户端机器名；2 字节整型客户端颜色深度；4 字节整型 RDP 通道数目；7 字节字符型 RDP 通道名；4 字节整型 RDP 通道标志。

服务器收到 MCS 初始化连接后，返回一个连接响应，类似于 MCS 初始化连接，微软在连接响应的 userdata 中加入了 RDP 服务器编码信息。数据包包括：1 字节整型 Result 值；19 字节 Target Domain Parameters；1 字节整型 connection id 值；23 字节 TAG_SRV_INFO；23 字节 TAG_SRV_SRV3；23 字节 TAG_SRV_CRYPT。其中，TAG_SRV_INFO 主要包括 2 字节整型 RDP 服务器版本。TAG_SRV_SRV3 的内容主要是实现 RDP5 通道数据到 MCS 通道数据的转换。TAG_SRV_CRYPT 主要包括：4 字节整型 RC4 密钥长度；4 字节整型加密级别（0X00000000 表示不加密；0X00000001 表示 40 位从客户端到服务器单向加密；0X00000002 表示 40 位双向加密；0X00000003 表示 128 位双向加密）；9 字节整型随机数 salt 长度；4 字节整型 RSA 信息长度；27 字节服务器 salt 值；4 字节整型 CA 证书长度；27 字节 CA 证书内容；4 字节整型服务器证书长度；27 字节服务器证书内容。

3.1.3　MCS EDrq

MCS 连接初始化完成后，客户端发送 MCS Edrq（Erect Domain request）到服务器。从我们截获的数据包分析得到 RDP 的 MCS Edrq 与 T 125 标准中的 MCS Edrq 数据包格式完全相同，详细格式可参考文献[6]第 10.6 节。

3.1.4 MCS AUrq 和 MCS AUcf

在客户端发送 MCS Edrq 后，客户端接着发送 MCS AUrq（Attach User request）。数据包主要包括客户端想加入域的用户名。服务器收到 MCS AUrq 后发送 MCS AUcf（Attach User confirm）用于用户的确认，数据包主要包含 user id。AUrq 和 AUcf 的详细格式可参考文献[6]第 10.15 和 10.16 节。

3.1.5 MCS CJRQ 和 CJCF

客户端收到 AUcf 后，再发送 MCS CJRQ（Channel Join ReQuest）用于建立通道加入请求。数据包包括：2 字节整型 user id；2 字节整型 channel id。

服务器收到 MCS CJRQ 后，返回配对的 CJCF（Channel Join ConFirm）用于确认通道的建立。数据包包括：1 字节字符型 RT_SUCCESSFUL；2 字节整型 user id；2 字节整型请求的 channel id；2 字节整型确认的 channel id。其中请求的 channel id 值和确认的 channel id 值应相同。

3.2 会话阶段

会话阶段包括两种类型数据包，一种为类似于建立阶段的 TPKT_TPDU_MCS 结构的数据包，另一种为微软自行设计的 RDP5 数据包，客户端发出的都是 TPKT_TPDU_MCS 结构的数据包，而服务器发送的主要是 RDP5 数据包。RDP5 数据包无 TPKT、TPDU 和 MCS 结构，而用更简单的头信息代替。RDP5 数据包包括：1 字节整型开始信息（0X80 表示加密数据包；0X00 表示非加密数据包）代替以 TPKT_TPDU_MCS 结构开始的 0X03；3 字节整型数据长度；14 字节字符型加密信息（如果是加密数据包）；1 字节整型数据包类型（类型值及意义见表 1）；多字节数据。

从服务器发出的数据包内容主要是 orders，用于描述客户端绘制各种图形的指令，orders 的详细定义请参考文献[7]。从客户端发送的数据包内容主要是键盘和鼠标事件。

表 1　RDP5 数据包类型

标识值	类型
0X00	Orders
0X01	Bitmap
0X02	Palette
0X05	NullSystemPointer
0X06	DefaultSystemPointer
0X07	MonoPointer
0X08	MousePosition
0X09	ColorPointer
0X0a	CachedPointer
0X0b	Pointer

4 基于 Linux 的 RDP Server

在获得 RDP 数据包和参考开放源码的 Rdesktop 基础上，设计开发了一种基于 Linux 的 RDP Server（RServer）。RServer 作为 Linux 上的终端服务器，能实现从基于 Linux 的 Rdesktop 或基于 Windows 的终端客户端（远程桌面连接）连接终端服务器，从而提供一种使用 RDP 协议的软件资源共享平台，使得 Linux 终端用户和 Windows 终端用户通过 RServer 可共享多个 Windows 终端服务器上的软件资源。

4.1 RServer 系统结构

RServer 运行后，启动一个主进程和一个会话管理进程。主进程监听客户端连接，若有连接则创建一线程负责客户端与 RServer 连接初始化、RDP 数据包处理、位图缓冲等业务。在处理连接时采用多线程技术，即多个客户端连接时每个客户端对应一线程。每个线程对应一个会话服务，会话服务通过唯一的会话标识符标识。会话管理进程基于会话标识符实现对不同用户的会话管理。如图 3 中 Rserver 部分，RServer 主要包括连接初始化模块、会话管理模块、RDP 数据包处理模块、位图缓冲模块、打印重定向模块等，程序在 Fedora 4 操作系统上，采用 C 语言开发。

（1）初始化连接模块。当 RServer 启动时的主进程启动后调用连接模块，连接模块通过调用 Socket 的 listen 和 accept 函数监听客户端连接模块发出的连接请求，若有连接主进程则创建一线程完成 TCP/IP 连接初始化，建立正常连接。

建立正常连接后，采用公钥加密体制的随机数分配方式，生成用于数据加解密的 RC4 分组密码算法的会话密钥。主要过程包括：①RServer 产生一个 32 字节随机数 S_random 和一对密钥{PK_S，SK_S}，并向客户端发送 S_random||PK_S||IDS；②客户端产生一个 32 字节随机数 C_random，根据 S_random 和 C_random 生成 RC4 算法的会话密钥，并向 RServer 发送 E_{PK_S}[C_random]||IDC；③RServer 通过 D_{SK_S}[E_{PK_S}[C_random]]得到 C_random，根据 S_random 和 C_random 生成 RC4 算法的会话密钥。至此，RServer 和客户端均已获得数据通信的会话密钥。

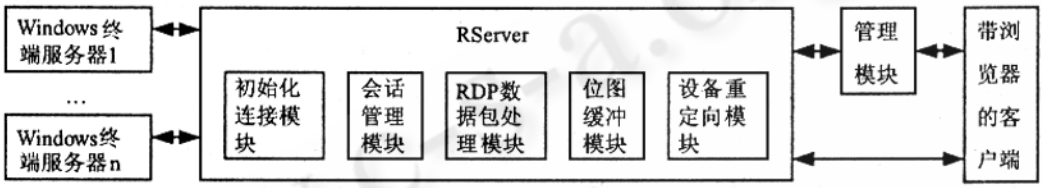

图 3　RServer 系统结构和系统部署图

（2）会话管理模块。当客户端用户初次登录 RServer 时，对应的线程会请求会话管理进程创建一个对应的会话服务线程。一个会话服务对应的数据结构信息有：会话标识符、会话用户名、会话桌面分辨率、会话建立时间、会话断开时间、会话服务状态。会话服务状态包括忙状态和断连状态。会话服务线程再进行会话的 X 环境初始化，并将会话服务状态设置为忙状态。

当用户选择注销会话时，会话管理进程结束对应的会话服务线程，回收内存资源。

当用户选择断开会话时，对应的会话服务线程继续运行，将会话服务状态设置为断连状态，RDP 数据包处理模块、位图缓冲模块等停止向客户端输出显示信息。

当用户断连即用户在上次断开会话后再次连接 RServer 时，会话管理进程在处于断连状态的会话服务表中查找对应用户的会话服务，再将会话服务状态设置为忙状态，并向客户端输出用户断开时的显示状态。

（3）RDP 数据包处理模块。RDP 数据包处理模块参考 Rdesktop 设计，从层次上分，分 TCP 层、ISO 层、MCS 层、安全层、RDP 应用层实现 RDP 数据包处理，如建立阶段的 ISO DP 8073 连接建立数据包、MCS 连接初始化数据包、MCS Edrq 等，会话阶段的 RDP5 格式数据包等。接收、解析从客户端提交的 TPKT_TPDU_MCS 结构的鼠标键盘数据包信息。接收从终端服务器返回的 RDP5 格式的 orders 等。

（4）位图缓冲模块。位图缓冲模块主要以槽的方式缓存以终端服务器发送过来的 orders 指令形成的位图数据，在缓存的同时，生成相应位图的 CRC，通过比较位图的 CRC 来判断客户端的位图与缓冲区中的位图是否一致，从而决定是否需要在客户端绘制位图。这种方式比不经过缓冲而直接绘制的方式要节省更多的网络带宽，尤其是位图相同或类似的情况下。

（5）设备重定向模块。采用终端服务模式的所有数据都存放在终端服务器中，如果用户需使用本地打印机打印输出存放在服务器上的数据，就需要使用设备重定向把客户端的打印机设备映射到服务器，在服务器端对应的是一个“会话打印机”。在打印文件的时候，文件内容被服务器上的打印机驱动程序转换为 PostScript 格式的打印文件，然后分解为多个 IRP（I/O Request Packet）包通过虚拟通道发送到客户端，实现本地打印功能。

除打印重定向外，其它还包括磁盘重定向和剪贴板重定向。磁盘重定向将客户端硬盘映射到终端服务器上，实现了在终端服务器上即可浏览客户端硬盘并进行文件操作的能力。剪贴板重定向实现了终端服务器和客户端的复制/粘贴。

4.2　实现网络化制造的软件资源共享

在区域网络化制造领域，强调企业间的协作和区域内的软件资源共享，以此达到提高企业产品设计的创新能力，实现产品设计制造的低成本和高效率的目标。构建软件资源共享服务，是实现网络化制造的重要保障。使用 RServer 可构建软件资源共享服务平台，使得用户端只需要配置标准浏览器即可安全地共享使用在数据中心安装在终端服务器群上的多种软件资源。如产品开发方面的 CAD、CAE、CAPP、CAM、PDM、

网络化协同产品设计等软件;在经营管理方面的 OA、ERP、SCM、CRM 等软件。

系统部署如图 3 所示,终端服务器群处于内网,安装 Windows Server 2003 操作系统、终端服务组件、应用系统(如 CAD、CAE、CAPP)等;RServer 运行于 Fedora 4 操作系统,具有内网和外网地址;管理工具处于外网,用 JSP 结合 Derby 10.0 数据库管理系统开发,主要实现用户管理、用户组管理、终端服务器管理和终端服务器上的应用程序发布等功能,需要在 Apache Web 服务器上运行。客户端处于外网,要求安装 Java 运行环境,配置标准浏览器。同时我们利用 Java Applet 技术开发了应用于客户端的连接 RServer 的 JavaRDP,使得客户端只需具备 Java 运行环境和浏览器,即可通过 Rserver 共享多个 Windows 终端服务器上的软件资源。客户端共享终端服务器上软件资源的工作流程如下:

(1) 客户端用户启动浏览器访问管理工具所在的 Web 服务器,若初次访问,浏览器自动下载 JavaRDP。

(2) 输入用户名、密码成功登录后,看到的是该用户能使用的应用软件链接。

(3) 用户单击相应应用软件链接即可通过 RServer 执行终端服务器上的应用软件。应用软件的执行均在终端服务器上完成,运行后的界面信息返回到客户端的 JavaRDP,再显示输出,客户端用户输入键盘、鼠标信息经 RServer 到终端服务器。

5 结束语

本文利用中间人攻击的原理,获得了微软具有商业秘密应用于终端服务的 RDP 协议数据包格式,由此,开发出了基于 Linux 的 RDP 服务器 RServer。经测试,RServer 与 Windows 终端服务器通信正常。使用 RServer 可构建软件资源共享平台,实现客户通过 Rserver 共享多个终端服务器上软件资源的目的。实际上,作为一种软件资源共享平台,RServer 可广泛应用于具有分支机构、需要远程数据处理的企事业,具有较好的市场前景。

参考文献

1 Mattew Chapman, et al. Rdesktop[CP/OL]. http://www.rdesktop.org, accessed May 2006.

2 Erik Forsberg. Microsoft Terminal Services vulnerable to MITM – attacks[EB/OL]. http://www.securityfocus.com/archive/1/317244, Apr 2003.

3 Erik Forsberg, et al. rdpproxy[CP/OL]. http://www.cse.unsw.edu.au/~matthewc/rdesktop/rdpproxy, accessed June 2006.

4 Alex McKenzie, et al. ISO Transport Protocol Specification, RFC905[S], Apr 1984.

5 Y. Pouffary, et al. ISO Transport Service on top of TCP (ITOT), RFC2126[S], Mar 1997. [6] International Telecommunications Union. Multipoint Communication Service Protocol Specification, T125[S], Nov 1993.

6 Microsoft, et al. Application Sharing, T128[S], Mar 1997.