

文章编号:1001-9081(2006)03-0739-03

一种动态字符录像数据的关键字搜索算法

史海峰,徐 涛

(南京航空航天大学 信息科学与技术学院,江苏 南京 210016)

(shfnoopy1228@163.com)

摘 要:从 Telnet 协议工作原理,分析了字符终端录像数据的录像及播放原理。根据录像数据的动态特性,提出了一种在字符录像数据动态播放的过程中,对字符录像数据进行关键字搜索的算法,并从字符串获取和字符串匹配算法的选取两方面对算法进行改进。通过时间测试证明了算法的正确性及实用性。

关键词:关键字搜索;字符串匹配;Telnet 协议;控制码

中图分类号: TP391.1 **文献标识码:** A

Keyword searching algorithm about dynamic character record data

SHI Hai-feng, XU Tao

(College of Information Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing Jiangsu 210016, China)

Abstract: Based on the working principle of Telnet protocol, the record and play mechanism of the record data about terminal was analyzed. According to the dynamic speciality of record data, a keyword searching algorithm was introduced working in the process of playing the record data. And the algorithm was improved from getting string and choosing appropriate matched string. Time test proves its correctness and practicability.

Key words: keyword searching; string matching; Telnet protocol; control-code

0 引言

目前具有相对简单配置和易维护性的字符终端设备在国内一些关键业务领域(如银行、证券以及政府等部门的业务处理平台)中应用仍然广泛,为防范操作风险,保障信息安全,考虑到这些行业对安全审计的要求,基于字符终端的监控系统应运而生。通常这类监控系统通过特定的采集机制,对终端操作行为进行数据的采集和存储,并可以通过回放和定位采集数据来审计终端操作行为,本文中称采集的数据为录像数据,由于是对字符终端的监控,这类录像数据都是字符数据。在基于安全审计的监控系统中,经常需要在指定范围(比如特定时间和地点等)内进行录像信息的搜索和定位,或支持录像数据的联动报警,甚至一些监控系统为支持备份/恢复和数据挖掘模块的功能,都涉及到在录像数据中进行关键字信息搜索。

基于字符终端监控的录像数据有如下两个特点:

1) 录像数据是海量数据,存放在数据库表中,当监控系统中录像的字符数据满足一定大小,就根据录像时间和终端号等形成一条录像记录存放在表中。

2) 录像数据是一种动态的字符数据,录像数据中的字符分为两类,普通字符和控制码字符,前者为终端屏幕实际显示的字符信息,后者是对终端操作行为的描述。

考虑在录像数据中搜索某关键字,由于关键字可能被拆分在表中不同的记录中,搜索时至少需要组合两条录像记录。由于录像数据是一种动态的字符数据,直接对组合的录像记录搜索并不能满足搜索要求。假设当前在对某一字符终端进行录像,终端用户进行登录操作,登录名为“Admin”,在用户

输入无误情况下,入库的录像数据为:

```
login: Admin
Password: * * * * *
```

此时在录像数据中搜索“Admin”这个关键字,可以直接搜索到指定信息。但若用户在输入“Admin”时发生错误,比如输入“Admo”后发现错误,用返回键消去字母“o”,再输入“in”,监控系统会使用一种称为控制码的特殊字符记录该返回键动作,假设返回键在当前字符终端的控制码为“####”,则此时录像数据为:

```
login: Admo#### in
Password: * * * * *
```

此时直接对录像数据搜索关键字“Admin”,将搜索不到该字符信息。

类似返回键的还有上、下、左、右移动键,回车键,删除键,插入键以及其他的一些特殊键,使用这些特殊键,在录像过程中,将通过相应的控制码记录字符处理过程,因此关键字可能会被控制码分离,而不是一个字符串整体。

本文中的录像数据是对特定网络内所有字符终端的操作过程的监控数据,录像数据可以看作是网络内所有 Telnet 会话数据。本文首先分析了 Telnet 协议工作原理及基于该协议的监控原理,然后通过一个字符播放控件,在该控件动态处理录像字符数据的过程中,提出了一种对该字符录像数据的关键字搜索算法,并对该算法进行改进,最后对该算法进行时间测试。

1 录像及播放原理

1.1 Telnet 协议工作原理

Telnet 是一种 C/S 模型的服务,它实现了基于 Telnet 协

收稿日期:2005-09-07 修订日期:2005-12-27

作者简介:史海峰(1981-),男,江苏海安人,硕士研究生,主要研究方向:信息安全、监控系统;徐涛(1962-),男,四川江油人,教授,主要研究方向:计算机视觉、图像处理、信息安全。

议的远程登录, Telnet 协议定义了一种交互的、面向字符的通信, 它定义了数据和命令在网络中的传输方式, 同时定义了一种通用字符终端: 网络虚拟终端 (Net Virtual Terminal, NVT)。NVT 是一种虚拟设备, 连接的双方都要把自身的终端特性映射成该虚拟终端的描述, 即通过把通信数据和 NVT 相互转换来进行交互^[1,2]。

对于从本地发送的数据, 当键入普通字符时, NVT 将按照其原始含义传送到远地机器; 当键入特殊键 (包括组合键) 时, NVT 将其理解为控制命令, 把它转化为对应的控制码在网络上传送, 并在其到达远地机器后转化为相应的控制命令。从远地返回的数据的过程类似。引入 NVT, 同时将普通字符与控制码区分, 保证了客户机与服务器间传送所有可能的普通字符以及所有控制功能; 同时客户机可以无二义性地指定信令, 而不会发生控制功能与普通字符的混乱。

1.2 录像工作过程

Telnet 协议是 TCP/IP 协议族中的一员, 所以本地与远程主机建立的连接是一个 TCP 连接, 录像过程是通过采集探针, 获取本地主机和远程主机之间传送的 IP 数据报, 通过解析和过滤该数据包, 得到普通字符以及控制码, 根据一定的存储策略形成录像数据。

本文中受控字符终端的主机操作系统为 Unix/Linux, 采集探针可以重新编译新的 login 程序。新的 login 程序需要派生出子进程, 调用原操作系统的 login 程序 (需修改名称, 且存放路径不变); 新的 login 进程需要做 I/O 重定向操作, 不影响原来登录操作, 包括不干预用户名和密码的输入与验证操作, 沿袭原系统的访问安全; 同时新 login 进程需要多占用一个伪终端设备资源, 进程通过访问伪终端设备来获取字符终端的标准输入、输出数据, 记录字符终端操作过程, 通过对获取数据的解析和过滤, 形成录像字符数据, 满足一定大小的录像数据保存入库。

1.3 播放原理

为确保搜索到关键字, 搜索时必须结合录像机制, 模拟字符终端的字符显示过程, 在字符显示过程中动态搜索关键字。模拟字符终端显示过程可以看作是播放录像数据, 这里采用了一个 TEmulVT 控件来播放录像数据。这是开源代码 FTerm 中的一个控件, TEmulVT 代码中包含了基本的 Telnet 功能, 可对 Ansi, SCOANSI, FTTERM, VT100 和 VT220 等多种终端类型的字符序列进行解析。该控件提供一个 24 行 80 列的屏幕视图, 通过读取录像字符数据中每一个字符, 将其中的普通字符在屏幕视图上正常显示, 同时解析录像数据中的控制码, 进行控制码对应的字符操作, 这样便可以在屏幕视图中模拟字符终端屏幕的显示过程。控件在分析一个字符时, 可以取得当前屏幕中的所有字符串以及光标所在位置。

2 搜索算法

2.1 搜索思想

搜索基于这样的过程: 每读取数据库中一条录像记录, 分析录像记录中每一个字符, 通过控件处理完当前字符, 获取当前屏幕数据, 再判断屏幕数据中是否有关键字, 如果有则返回搜索结果, 若没有, 则继续分析下一个字符。当一条录像记录分析完毕, 继续读取下一条录像记录, 当所有录像记录都分析完毕, 搜索过程结束。

若分析的字符是控制码一部分时, 则不需要比较, 继续读取字符, 直到取得完整的控制码字符, 解析该控制码, 同时通

过控件执行控制码对应的字符操作。若控制码被分割在一条录像记录结尾和下一条录像记录开头, 需要将当前录像记录的结尾字符放到下一条录像记录开头再进行分析, 算法中, 只要当前录像记录中一定长度的尾部字符串中含有控制码字符, 则将该尾部字符串放到下一条录像记录的开头。

对于大量录像数据, 考虑到搜索中的数据交换、读、写等操作, 如果直接读取每条录像记录, 并且分析每个字符, 搜索效率将非常低。判断屏幕数据中是否有关键字, 是一个字符串匹配过程, 假设录像记录有 10 万条, 一条录像记录长度为 1920 个字符 (24 行 80 列), 假设平均搜索 5 万条录像记录, 每条记录中有 1000 个字符需要进行关键字匹配, 这样就需要进行 5000 万次字符串匹配。若称关键字为模式串, 选取用来和模式串匹配的录像字符串为文本串, 下面主要从文本串的获取来改进搜索过程。

2.2 搜索改进

对文本串获取的改进主要从两方面考虑: 减少匹配次数和减少文本串长度, 根据算法的流程, 从以下三个方面进行改进, 其中 1)、2) 是基于减少匹配次数, 3) 是基于减少文本串长度。

1) 对关键字进行分类, 将关键字分为非动态关键字和动态关键字。非动态关键字为无需人工交互而在屏幕上显示的字符, 这些字符在保存入库时不会被控制码拆分; 而动态关键字则为需要经过交互输入的字符。这里涉及到一些领域知识, 比如在银行领域, 监控每一个银行业务终端, 通常终端名、系统时间等字符的显示是动态生成的, 搜索这些字符串时, 可认为是非动态关键字, 而对用户办理金额、业务类型等需人工参与输入的字符信息则为动态关键字。在实际搜索关键字过程中, 如果不能确定关键字类型, 则认为是动态关键字。

对于非动态关键字的搜索, 直接通过 SQL 语句就可以完成, 通过数据库系统执行 SQL 语句的速度是非常理想的, 特别是在含有大量录像数据的数据库中搜索。下面 2)、3) 的改进是针对动态关键字的搜索。

2) 在动态关键字搜索中, 每读入一条录像记录时, 判断是否应该对当前录像记录或下一条录像记录进行搜索, 分两种情况: 第一, 判断当前录像记录是否有关键字中的每一个字符, 如果是, 则该记录可能有关键字, 需要对当前录像记录进行分析; 第二, 针对可能会出现“关键字被分割在一条录像记录结尾和下一条录像记录开头”和“控制码被分割在一条录像记录结尾和下一条录像记录开头”两种情况, 取当前录像记录一定长度的尾部字符串, 若该字符串中含有关键字中的字符或含有控制码字符, 则需要对下一条录像记录进行分析, 且将该尾部字符串加入下一条录像记录的开头。

3) 在需要对当前的录像记录进行搜索时, 需要对录像记录中每个字符进行分析, 为减少比较字符的长度, 首先判断当前字符是否为关键字中的某一个字符, 若不是, 分析录像记录中下一个字符, 若是, 则取当前字符前后各关键字长度的字符串 (之所以要取当前字符后面的关键字长度的字符串, 是考虑到终端用户通过 Tab 键和上、下、左、右移动键等来修改字符信息的情况, 在这些情况下, 当前字符不是屏幕最后一个字符), 用该字符串和关键字进行比较, 若不含关键字, 继续分析录像记录中下一个字符, 若有关键字, 返回搜索结果。

2.3 算法描述

搜索算法运行在一个搜索线程中。在创建搜索线程的同时, 创建一数据获取线程, 该线程设置一个缓冲区 BufList。从数据库中读取多条指定搜索范围内的连续的录像记录, 放到

BufList 中。搜索线程每次直接从 BufList 中读取一条录像记录,通过缓冲区解决了搜索和数据传输的同步问题,使搜索线程和录像数据获取线程能够并发运行,提高了程序效率。

约定变量 KeyWord 为搜索关键字,RecordID 为当前的录像记录号,RecordData 为当前录像记录字符串,nTailLength 为取录像记录尾部字符串的长度,sLastTail 和 sNowTail 分别表示上条和本条录像记录的尾部字符串。

算法描述如下(Object Pascal 语言):

```

if KeyWord 为非动态关键字 then
begin
    执行 SQL 语句 'select * from RecordTab where RecordData like
    % KeyWord%';
    返回 SQL 语句搜索结果;
end else
begin
    sLastTail: = '';
    取缓冲区 BufList 中的当前录像记录号 RecordID;
    While( RecordID > 0 ) do
    begin
        获取当前录像记录数据 RecordData;
        bNoKeyWord: = true;
        RecordData: = sLastTail + RecordData;
        if RecordData 中含有 KeyWord 中每一个字符 then (1)
            bNoKeyWord: = false;
        sNowTail: = RightStr( RecordData, nTailLength );
        //取尾部字符串;
        if ( sNowTail 中不含控制码字符 ) and
            ( sNowTail 中不含有 KeyWord 中任一字符 ) then (2)
            sNowTail: = '';
        if ( bNoKeyWord = false )
            //当录像记录中含有每个关键字字符或上一条记录
            or ( sLastTail < > '' ) then
                //尾部字符串含有关键字字符或控制码字符时
        begin
            i: = 1;
            nLength: = Length( RecordData ) - Length( sNowTail );
            While i < nLength do
            begin
                ch: = RecordData[ i ]; inc( i );
                if ch 为控制码开头字符 then
                begin
                    repeat sCtr: = ch + RecordData[ i ]; inc( i );
                    until sCtr 是完整的控制码;
                    由控件执行相应控制码的字符操作;
                    continue;
                end;
                if KeyWord 中含有字符 ch then (3)
                begin
                    通过控件获得当前字符所在光标位置;
                    根据光标位置获得屏幕中当前字符前后各关键字长
                    度的字符串 S;
                    if S 中含有 KeyWord then (4)
                        返回搜索结果;
                    end;
                end;
            end;
            sLastTail: = sNowTail;
            //将当前的尾部字符串转为下一条记录的开头;
            取下一条录像记录 RecordID;
        end;
    end;
end;

```

算法中函数 Length(s) 返回字符串 s 的长度,RightStr(sData, nLen) 返回字符串 sData 的右边 nLen 长度的字符串。

2.4 匹配算法选取

本文中的字符串匹配基于准确的单模式匹配,目前这类匹配算法有很多,包括 BF 算法、KMP 算法、BM 算法及它们的一些改进算法。匹配算法在匹配过程中遇到不匹配的字符时,需要将模式串向后移动,移动多少就是各种算法的不同之处。BF 算法是一种最简单但效率比较低的算法,遇到不匹配的情况直接向后移动一个字符位置;KMP 算法根据模式串计算一个 next 数组,遇到不匹配字符时利用已匹配的信息结合 next 数组来移动;BM 算法首先根据字符集计算所有字符在模式串中的位置,并且算法从模式串的最右端进行反向匹配,并根据已经匹配的部分来确定移动量。

在改进的算法中,标出的(1)~(4)处需要进行字符串匹配,这里涉及到两种类型的字符串匹配情况:从一个字符串中匹配一个字符,(1)~(3)属于这一类;在 $2 * Len + 1$ 长度的字符串中判断是否含有 KeyWord,其中 Len 是 KeyWord 的长度。两类匹配比较简单,这里仅从 BF、KMP 和 BM 算法中选取匹配算法,不考虑它们的改进算法。

对于第一种情况,模式串就是一个字符,使用 BF、KMP 和 BM 算法在匹配算法中的字符比较次数完全相同,而 KMP 和 BM 算法在每次匹配前需要进行预处理,占用额外的资源,所以使用 BF 算法。

对于第二种情况,在海量的录像数据中,实际搜索的大部分情况下字符将不匹配。在已知字符不匹配的前提下,使用 BF 和 KMP 算法需要比较 Length(KeyWord) + 1 次字符,由于 BM 算法是反向匹配,大部分情况下它只需要比较 2 次,所以使用 BM 算法;且 BM 算法的预处理已经计算了所有字符在模式串中的位置,算法中标出的(3):判断 KeyWord 中是否含有字符 ch,可以直接判断。

2.5 算法说明

考虑到算法的简洁性,许多地方没有给出详细实现过程,也没有体现异常处理,另外算法可以从以下三个方面完善:

1) 如果一个用户始终在多次重复进行“输入-返回键-修改-输入-返回键-修改”的过程,则在理论上,这种录像数据字符可能分布在几条录像记录中,所以如果在一条录像记录中检测到连续的控制码,需要特别处理,在某些特定领域,甚至可认为该过程为可疑行为,需要单独处理。

2) 对空格和中文字符的处理:动态搜索中遇到空格,不需要进行比较,并判断下一个字符是否为空格,直到下面出现非空格字符;由于中文占两个字符,所以出现中文首字符,需要合并下一个字符再处理,在实际应用中这两种情况都比较多。

3) 搜索时加入先验知识,比如用户登录时,必须经过用户名和密码输入,因此在搜索“Admin”用户名的登录记录时,可以同时加“login”或“Password”两个条件,这两者都是非动态关键字,搜索速度会提高很多。

3 算法效果

本文录像数据以 Mysql 数据库中的 Text 字符串格式保存,算法通过 Delphi 8.0 实现,运行环境为:Windows 2000 Server 操作系统,主频 600MHz Intel Celeron CPU,内存 256 + 128M。

(下转第 745 页)

结构作了详细规定,将文件分为专有文件(Dedicated file)和基本文件(Elementary file)。其中,专有文件可以看成是 Unix 文件系统上的“目录”,基本文件可以看成是“文件”。基本文件进一步分为透明文件、线性变长文件、线性定长文件和循环文件。com.bestsoft.javacard.filesystem 包中文件类的继承结构^[4]、关键属性和方法如图5中所示。

从图5可以看出,File类是所有文件对象的父类,类DedicateFile和ElementaryFile分别对应专有文件与基本文件,它们直接继承自File类。文件系统类FileSystem的对象可以看作Unix文件系统上的根目录,因此,将类FileSystem作为DedicatedFile类的子类。类TransparentFile和LinearVariableFile分别对应透明文件与线性变长文件,它们继承了ElementaryFile类。线性定长文件可以看成是线性变长文件的特殊情况,所以,将LinearFixedFile类作为类LinearVariableFile的子类。同样,循环文件可以认为是一种特殊的线性定长文件,因此,CyclicFile类继承了LinearFixedFile类。

4 结语

本文根据Sun公司的Java Card 2.2.1规范,深入研究了Java卡的关键技术,提供了Java卡的一种具体实施。文中详细描述了Java卡系统所采用的系统构架和 workflow,深入分析了Java卡虚拟机与API等关键技术,并讨论了为提高Java卡虚拟机的执行效率所采用的框架叠加、非持久对象临时化、异常对象全局化及查表解释执行等策略。该Java卡系统共占用39.8K空间,其中,卡操作系统和Java卡运行时环境(除API外)共占用32.1K字节,标准及扩展API占用7.7K字节。为了测试Java卡的性能,根据Java卡应用编程指南^[8],编写了一个符合GSM(Global System for Mobile communications)

11.11规范的Java卡应用。测试结果表明,该GSM应用能够在Java卡上正确高效地运行。

Java卡虚拟机采用解释执行方式,所以执行速度相对较慢。进一步优化Java卡系统的性能以及提高系统的安全性,是今后的研究重点。

参考文献:

- [1] Sun Microsystems, Inc. Virtual machine specification, java card platform, version 2.2.1 [EB/OL]. <http://java.sun.com/products/javacard/index.jsp>, 2003.
- [2] International Standard ISO/IEC 7816-4, Identification cards. Integrated circuit (s) cards with contacts, Part 4: Interindustry commands for interchange[S], 1995.
- [3] Sun Microsystems, Inc. Runtime environment specification, java card platform, version 2.2.1 [EB/OL]. <http://java.sun.com/products/javacard/index.jsp>, 2003.
- [4] 王克宏,徐剑军,徐鹏. Java 嵌入技术[M]. 北京:清华大学出版社,1998.
- [5] 探砂工作室. 深入嵌入式Java虚拟机[M]. 北京:中国铁道出版社,2003.
- [6] LINDHOLM T, YELLIN F. Java 虚拟机规范[M]. 玄伟剑,译. 北京:北京大学出版社,1997.
- [7] Sun Microsystems, Inc. Application programming interface, java card platform, version 2.2.1 [EB/OL]. <http://java.sun.com/products/javacard/index.jsp>, 2003.
- [8] Sun Microsystems, Inc. Application programming notes, java card platform, version 2.2.1 [EB/OL]. <http://java.sun.com/products/javacard/index.jsp>, 2003.
- [9] 刘嵩岩,毛志刚,叶以正. Java卡的研究与实现. 微电子学[J]. 2000,1(6):402-405.

(上接第741页)

测试针对两种算法:一种是采用原始的搜索思想,没有从文本串的获取和字符串匹配算法的选取方面进行优化,第二种则是改进后的搜索算法。在搜索程序运行过程中,数据库的访问、数据交换以及系统本身的运行会占用大量的资源,因此算法测试只针对搜索时间,测试是在2万条录像记录中进行的。由于关键字中的字符在录像数据出现的频繁度对搜索效率影响比较大,因此将动态关键字分为频繁动态关键字和非频繁动态关键字,前者表示该关键字中的字符在录像数据中出现频繁的动态关键字。根据关键字的性质,测试分5组进行,测试结果如表1。

从表中可以看出如果关键字中字符出现比较频繁,改进后的算法搜索时间能提高6倍左右;若关键字中字符出现不频繁,第一种算法的搜索算法耗时基本没有变化,用改进后的算法,搜索时间通常能提高50倍以上。

表1 算法平均搜索时间

算法关键字性质	改进前	改进后
非动态关键字	/	3~4s
第1000条记录中出现的频繁动态关键字	4~5min	44~60s
第1000条记录中出现的非频繁动态关键字	4~5min	≤10s
第10000条记录中出现的频繁动态关键字	40~45min	5~7min
第10000条记录中出现的非频繁动态关键字	40~45min	≤40s

4 结语

本文提出一种动态字符录像数据的关键字搜索算法,从文本串的获取和字符串匹配算法的选取两个方面对算法进行改进。在实际的字符终端监控系统应用中,搜索算法需要采用多线程技术,根据搜索范围及系统资源来分配多线程数及搜索任务,必要时应对搜索过的信息建立索引信息,为提高搜索效率,搜索过程中需考虑如何协调数据库工作,在数据交互时,还需考虑到数据的互斥问题。目前该算法已成功应用在一个银行字符终端监控系统中。

参考文献:

- [1] POSTEL J, REYNOLDS J. RFC854, Telnet Protocol Specification [S], 1983.
- [2] 杨冕,秦前清,吴娟娟,等. Telnet 协议在网络视频监控系统中的应用[J]. 计算机应用研究, 2005,22(4):159-161.
- [3] 陆晟,龚俭. 网络安全监测的集成管理[J]. 东南大学学报, 1999,29(5):11-15.
- [4] 李志淮,邢月华. 字符终端功能及其实现[J]. 小型微型计算机系统, 1995,16(3):48-51.
- [5] 陈国龙,陈火旺,康仲生. 基于内容的网络信息安全审计中的匹配算法研究[J]. 小型微型计算机系统, 2004,25(9):1676-1679.
- [6] 李雪莹,刘宝旭,许榕生. 字符串匹配技术研究[J]. 计算机工程, 2004,30(22):24-26.
- [7] BOYER RS, MOORE JS. A fast string searching algorithm [J]. Communications of the ACM, 1977,20(10):762-772.