基于行为关联分析的异常文件管理活动识别系统

彭国军[1,2]，王 滢[1+]，梁 玉[1]，于 慧[1]，王至前[1]

（1. 武汉大学 计算机学院 空天信息安全与可信计算教育部重点实验室，湖北 武汉 430072；
2. 中国人民大学 法学院，北京 100872）

摘 要：提出一种基于文件和网络行为关联分析的异常文件管理活动识别方法，并实现原型系统 F-Sensor。根据异常文件管理活动的时序特征，生成由 API 函数组成的行为序列；依据异常文件管理活动的实现特点，关联分析文件和网络行为以识别疑似异常的行为；结合白名单机制和异常文件管理行为间的依存关系，识别真正的异常文件管理活动。实验结果表明，F-Sensor 能较好地识别异常文件管理行为，误报率较低，利用其识别结果可检测恶意软件，有助于分析攻击者的控制意图。

关键词：异常识别；恶意软件；文件管理；系统调用；行为监控

中图法分类号：TP309.5 **文献标识号**：A **文章编号**：1000-7024（2015）12-3161-07

doi：10.16208/j.issn1000-7024.2015.12.001

Abnormal file management activities identification system based on association analysis of behaviors

PENG Guo-jun[1,2], WANG Ying[1+], LIANG Yu[1], YU Hui[1], WANG Zhi-qian[1]

(1. Key Laboratory of Aerospace Information Security and Trusted Computing, School of Computer, Wuhan University, Wuhan 430072, China; 2. Law School, Remin University of China, Beijing 100872, China)

Abstract: An approach to identify abnormal file management activities based on the association analysis of file and network behaviors was proposed, and the prototype system F-Sensor was implemented. Behavior sequences consisting of the API functions were generated according to the temporal characteristics of abnormal file management activities, and then file and network behaviors were associated to analyze and identify suspected abnormal behaviors based on implementation characteristics of abnormal file management activities. True abnormal file management behaviors were determined by combining white list and the dependency between the abnormal file management behaviors. Experimental results show that the method can well identify the abnormal file management behaviors with low rate of false positives. The identification result generated from F-Sensor can detect malware and make for the analysis of the attacker's control intention.

Key words: abnormity identification; malware; file management; system call; behavior monitoring

0 引 言

为保护敏感机密文件，文献[1，2]能准确识别基本的文件行为类型，但用户或淹没在庞大的监控和分析数据中，无法发现真正危险的文件行为；或仅能对个别敏感机密文件实施监控，不能全面识别系统中发生的危险文件行为。在软件行为检测领域，现有的动态检测技术[3-5]专注于程序的动态执行过程，但仅能简单地识别基本行为，且无法对单个行为定性（即正常或异常）；而静态检测技术[6]以程序的反汇编代码为分析对象，难以应对经过特殊处理的

收稿日期：2014-12-24；**修订日期**：2015-03-10

基金项目：国家自然科学基金项目（61202387、61373168、61202385）；中国博士后科学基金项目（2012M510641）；高等学校博士学科点专项科研基金项目（20120141110002）；武汉市青年科技晨光计划基金项目（201271031367）

作者简介：彭国军（1979-），男，湖北荆州人，博士，副教授，CCF 会员，研究方向为恶意软件检测、移动智能终端安全、电子证据；+通讯作者：王滢（1991-），女，河南许昌人，硕士研究生，研究方向为恶意软件检测、主机安全；梁玉（1988-），男，甘肃白银人，博士研究生，研究方向为恶意软件检测、移动智能终端安全；于慧（1992-），女，吉林延边人，本科，研究方向为恶意软件检测；王至前（1993-），男，辽宁大连人，本科，研究方向为恶意软件检测。E-mail：lasiawang@126.com

代码（如加密、混淆或变形），分析效率低下[7]，且存在同动态检测同样的问题[8]。

针对以上情况，为识别系统中异常的文件管理活动，本文在现有动态检测技术的基础上，以异常文件管理活动的时序特征和实现特点为依据，提出基于文件和网络行为关联分析的识别方法，并实现原型系统 F-Sensor。

1 问题描述

1.1 异常文件管理活动的定义

异常文件管理活动是指远程控制者通过网络连接到受控主机，进而借助控制主机（“控制端”）对受控主机（“被控端”）开展的文件管理操作。它主要包括远程浏览目录、远程搜索文件、远程上传文件（夹）、远程下载文件（夹）等行为（“上传/下载”在本文中相对于控制端而言）。本文拟对以上 6 种异常行为进行识别。

1.2 异常文件管理活动的时序特征

典型的控制行为是通过控制端向运行在受控主机上的被控端程序发送命令，被控端执行相应的操作并返回执行结果来实现的[9]，异常文件管理活动的工作模式也是如此。由于 CPU 的高速运转，被控端接收命令、执行操作和发送执行结果三者之间的时间间隔很小。而且，因控制端为人工操控，查看返回结果和发布新的命令都需要时间，故连续的管理行为之间也存在一定的时间间隔。因此，异常文件管理活动在时间上呈现同一行为内的网络和文件操作紧密相依，而不同行为间疏离的特点。

1.3 异常文件管理活动的实现特点

在具体实现时，以上 6 种异常文件管理行为又表现出不同的特征：

（1）远程浏览目录与远程搜索文件

被控端接收到浏览目录命令后，根据命令中指定的文件目录立即对其执行遍历文件操作，并将获取到的所有文件路径立刻发送给控制端。而对于远程搜索文件行为，被控端接收到搜索文件命令后，根据命令中指定的搜索目录立即递归地对该目录及其子目录执行遍历文件，最终将文件名或文件内容中包含搜索关键词的所有文件路径发送给控制端。

远程浏览目录本质上是一种特殊的远程搜索文件行为——当搜索关键词为“*”时对单层目录的搜索。

（2）远程上传文件与远程下载文件

被控端接收到上传文件命令后，即根据命令中包含的文件名称和内容创建并写入文件，全部写入后返回执行结果；对于远程下载文件行为，被控端接收到下载文件命令后，即根据命令中规定的文件名称，读取该文件内容并发送给控制端。

（3）远程上传文件夹与远程下载文件夹

被控端接收到上传文件夹命令后，首先根据命令中所包含的上传目标文件夹名称创建文件夹，然后根据后续传来的文件名称及内容创建并写入文件，如此递归直至上传目标文件夹中所有目录和文件均被创建和写入。

对于远程下载文件夹行为，被控端接收到下载文件夹命令后，即根据命令中规定的文件夹名称，递归地遍历和读取该目录及其子目录中的所有文件，并将文件内容发送给控制端。

1.4 相关概念的定义

本文以进程的特定网络和文件 API 调用记录为分析对象，根据异常文件管理活动的特征，分析和识别远程浏览目录、远程搜索文件、远程上传文件（夹）和远程下载文件（夹）这 6 种异常行为。为便于叙述，我们把分析过程中涉及的相关概念作如下定义：

（1）关键 API 函数：异常文件管理活动发生时所调用且可用于识别异常活动的 API 函数，包括遍历文件、读取文件、写入文件和创建文件夹等文件相关 API 函数，以及发送/接收数据等网络相关 API 函数。

（2）行为序列：由调用时间间隔小于一定值的相邻关键 API 函数所组成的 API 调用序列，即包含一个或多个完整文件管理行为的 API 调用序列。

（3）行为链表：用于存储行为序列的双向链表。

（4）分析结果序列：针对某一行为序列，记录其行为识别的中间和最终结果，包含识别出的行为类型、开始时间、结束时间和其它信息，如目标文件/目录、远程 IP/端口号、搜索关键词等。

（5）分析结果链表：用于存储分析结果序列的双向链表。

2 F-Sensor 系统及识别算法

2.1 系统架构

根据异常文件管理活动的特征，对异常文件管理行为进行识别，包括 3 个步骤：①拦截关键 API 函数，根据函数调用时间间隔划分行为序列；②根据关联分析算法，对每个行为序列进行网络和文件行为关联分析，提取疑似异常的文件管理行为；③结合白名单机制和异常文件管理行为之间的依存关系，剔除可疑行为中的误报行为，从而识别真正的异常文件管理行为。

基于此，本文实现了如图 1 所示的系统架构，其主要包含三大模块：行为监控模块、关联分析模块、行为识别模块，分别对应于以上 3 个步骤。

2.2 行为监控模块

行为监控模块利用 API Hook 技术，拦截系统中所有进程对关键 API 函数的调用，如 NtQueryDirectoryFile、NtReadFile 等，并将行为类型、调用时间、关键参数及由参数获取的重要信息记录到行为链表中，供关联分析模块分析。行为类型和记录信息的对应关系见表 1。

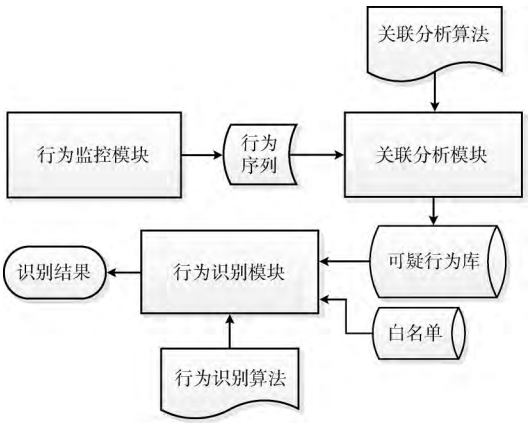

图 1　异常文件管理活动识别系统架构

完整行为序列的生成：行为监控模块在每个拦截函数中，检查当前调用时间与关键函数上一次调用时间的差，若大于一定的时间间隔（经实验分析取 1 s），则认为当前的行为序列已生成完整，随即创建新的行为链表来存储新的行为序列，并将本次调用插入新行为序列中；否则，将本次调用插入当前行为序列的末尾。此外，为方便分析，对同一行为序列中连续的遍历文件 API 调用、具有相同文件句柄的读取/写入文件 API 调用，以及具有相同套接字的发送/接收数据 API 调用进行合并。其中，遍历文件行为中可能包含多次调用对应的目录路径和文件名，而遍历目录在列表中的先后顺序代表了遍历顺序；遍历文件名是指遍历时指定的文件名称或包含通配符的字符串，对文件名的搜索关键词可在此参数中得以体现。

表 1　行为类型和记录信息

行为大类	行为类型	记录信息
文件行为	遍历文件	遍历目录路径列表、遍历文件名列表、返回 STATUS_NO_MORE_FILES 和 STATUS_SUCCESS 的次数之差、开始和结束时间
	写入文件	文件路径、写入数据长度、文件句柄、开始和结束时间
	读取文件	文件路径、读取数据长度、文件句柄、开始和结束时间
	创建文件夹	文件夹路径、开始和结束时间
网络行为	发送数据	发送数据长度、套接字、远程/本地 IP 和端口号、开始和结束时间
	接收数据	接收数据长度、套接字、远程/本地 IP 和端口号、开始和结束时间

2.3　关联分析模块

在行为监控模块工作的同时，关联分析模块一旦发现已生成完整的行为序列，即对该序列进行分析，识别出疑似异常的文件管理行为，并将最终的分析结果存储到可疑行为库中，以供识别模块查询分析。分析过程分为两步：

（1）以行为序列为分析对象，根据行为节点的不同类型，调用相应行为的识别算法以识别远程浏览目录、远程搜索文件、远程上传文件和远程下载文件 4 种行为，并将分析结果存储到分析结果链表中，如图 2 所示；

（2）以分析结果序列为分析对象，调用远程上传/下载文件夹行为识别算法以识别其是否存在远程上传/下载文件夹行为，若存在则对分析结果链表中与此行为相关的节点进行整合。

2.3.1　远程浏览目录和远程搜索文件行为的识别

这两种行为的识别在第一步分析过程中完成（如图 2 所示），在遍历行为序列时若遇到遍历文件行为，首先对行为序列中同属于一个遍历文件行为的离散节点进行整合，然后根据整合后节点中所包含的目录路径之间的关系，利用遍历文件行为类型识别算法判定整合节点的行为类型（搜索/浏览/未知），最后调用远程遍历行为识别算法，判定其是否属于远程遍历文件行为及其具体行为类型（远程搜索/远程浏览/远程未知）。

整合离散的遍历文件行为：有些搜索行为因涉及对许多目录下文件的遍历，耗时较长，在执行过程中可能会被其它行为（如网络收发行为）所截断，形成多个离散的行为节点。而搜索行为多是递归实现，对搜索目标目录及其每一子目录而言，遍历文件 API 函数以 STATUS_SUCCESS 返回值开始，直至该目录中文件遍历完毕时返回 STATUS_NO_MORE_FILES。因此，完整搜索行为的特征是返回 STATUS_NO_MORE_FILES 和 STATUS_SUCCESS 的次数之差为 0。据此整合行为序列中的离散行为节点。

遍历文件行为类型识别算法：该算法用于判断遍历文件行为的类型（见算法 1），根据遍历行为中所包含的目录路径之间的关系来确定。因有些浏览目录行为存在对遍历文件相关函数的多次调用，故若目录列表中全是同一目录，则将该行为判定为浏览目录。若目录列表中从第二个目录起所有目录的父目录都存在于其之前的目录集合中，则将该行为判定为搜索文件行为。若不符合以上两种情况，则判定为未知遍历行为。

算法 1：遍历文件行为类型识别算法

输入：行为序列中经过整合后的遍历文件行为 $a_i=<$

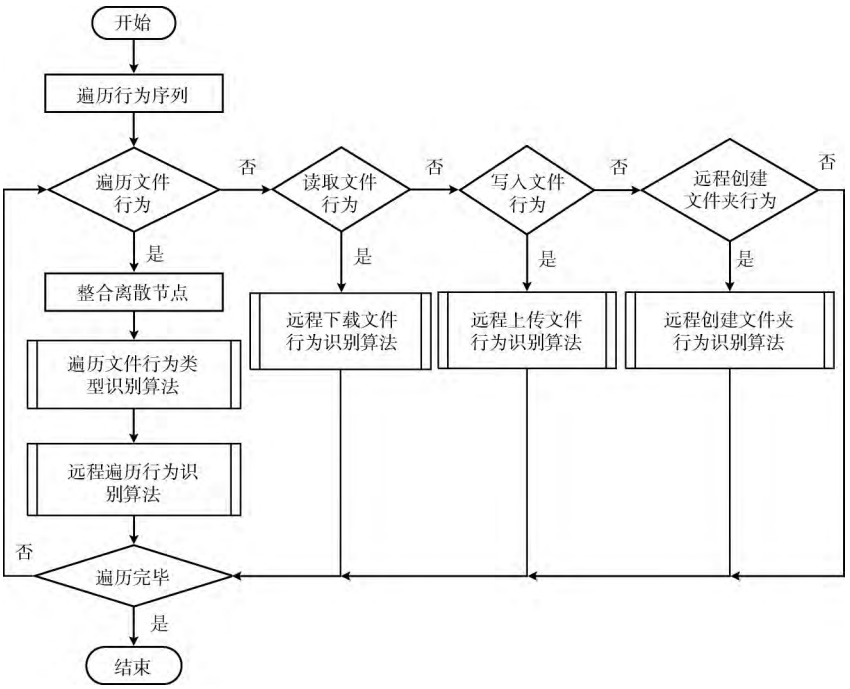

图 2 关联分析第一步的分析流程

beginTime，endTime，type，folderList，keywordList，nNoMoreFiles＞.

输出：遍历文件行为的类型 type＝＜BrowseFolder | SearchFile | UnknownTravese＞.

```
TraverseFileActJudge(ai) {
  bExist=FALSE; nIsnotFit=0;
  nEqualToFirst=0;
  for (int i = 1; i<folderList. size (); i++) {
    if (folderList [i]. fileName 与
      folderList [0]. fileName 相同)) {
        nEqualToFirst ++;
        bExist=TRUE;
      }
    else
      if (folderList 中 0 至 i-1 中存在
      folderList [i]. fileName 的父目录))
        bExist=TRUE;
      if (!bExist) nIsnotFit++;
  }
  if (nEqualToFirst==folderList. size () -1) {
    ai. type=BrowseFolder;
    return BrowseFolder;
  }
  if (nIsnotFit!=0) {
    ai. type=UnknownTravese;
    return UnknownTraverse;
  }
  else {
    ai. type=SearchFile;
    return SearchFile;
  }
}
```

远程遍历行为识别算法：该算法通过关联网络和文件行为来识别远程浏览目录/搜索文件行为，见算法 2。将遍历文件行为 a_j 判定为远程行为，需具备两个条件：①a_j 之前存在时间间隔小于一定值（经实验分析，本文取 500 ms），且属于小包的接收行为；②a_j 之后存在时间间隔小于一定值，且与满足条件的接收行为具有相同套接字的发送行为。

算法 2：远程遍历行为识别算法

输入：完整的行为序列 S＝｛a_i | i＝1，…，n｝，**遍历文件行为** a_j ＝＜beginTime，endTime，type，folderList，keywordList，nNoMoreFiles＞。

输出：分析结果序列 R＝｛r_i ＝＜beginTime，endTime，type，pArgList＞ | i＝1，…，m｝。

```
AnalysisBSFile (S, aj) {
  bRemote=false;
  for (int i=j-1; i>0; i--) {
    interval=aj. beginTime-ai. endTime;
    if (interval>500)   break;
```

```
        if（$a_i$ 是接收行为 &&$a_i$ 是小数据包）{
          for（int k=j+1；j<=n；j++）{
            interval=$a_k$. beginTime-
            $a_j$. endTime；
            if（interval>500）  break；
            if（$a_k$ 是发送行为 &&
             $a_k$. s==$a_i$. s）{
             r= {$a_i$，$a_j$，$a_k$}；R=r∪R；
             bRemote=true；break；
            }
          }
        }
        if（bRemote）break；
      }
    }
```

2.3.2 远程上传和下载文件行为的识别

这两种行为的识别在第一步分析过程中完成（如图 2 所示），在遍历行为序列时若遇到写入文件行为，则调用远程上传文件行为识别算法，判定其是否属于远程上传文件行为；若遇到读取文件行为，则调用远程下载文件行为识别算法，判定其是否属于远程下载文件行为。

远程上传文件行为识别算法：该算法通过寻找行为序列中写入文件行为之前满足条件（即二者时间间隔小于一定值且接收数据与写入数据长度大于 1/2）的接收行为，来识别远程上传文件行为，见算法 3。

算法 3：远程上传文件行为识别算法

输入：完整的行为序列 S= {a_i | i=1，…，n}，写入文件行为 a_j=<beginTime，endTime，type，filename，writtenLen>。

输出：分析结果序列 R= {r_i =< beginTime，endTime，
type，pArgList> | i=1，…，m}。

```
AnalysisUploadFie（S，$a_j$）{
  for（int i=j-1；j>0；j--）{
    interval=$a_j$. beginTime - $a_i$. endTime；
    if（interval>500）  break；
    if（$a_i$ 是接收行为 &&$a_i$. len>=$a_j$. len/2）{
    r= {$a_i$，$a_j$}；R=R∪r；break；}
  }
}
```

远程下载文件行为识别算法：与算法 3 类似，该算法遍历行为序列中读取文件行为之后的所有行为，寻找满足时间间隔小于一定值且发送数据与读取数据长度大于 1/2 的发送行为：若找到，则说明是远程下载文件行为。

2.3.3 远程创建文件夹行为的识别

该行为的识别在第一步分析过程中完成（如图 2 所示），在遍历行为序列时若遇到创建文件夹行为，则调用远程创建文件夹行为识别算法，判定其是否属于远程创建文件夹行为。

远程创建文件夹行为识别算法：该算法与算法 2 类似，即以行为序列和创建文件夹行为作为输入，通过关联文件行为前后的网络行为来识别远程创建文件夹行为。识别该行为是作识别远程上传文件夹行为之用。

2.3.4 远程上传和下载文件夹行为的识别

在完成对上述行为的识别之后，调用远程上传/下载文件夹行为识别算法，对分析结果序列中的远程创建文件夹和远程上传文件行为，或远程浏览目录/搜索文件和远程下载文件行为进行关联分析，判定是否存在远程上传文件夹或下载文件夹行为。

远程上传文件夹行为识别算法：该算法通过关联分析第一步得到的分析结果序列中存在的远程创建文件夹和远程上传文件行为，来识别远程上传文件夹行为。根据远程上传文件夹行为的实现特征，我们得知若远程上传文件行为之前存在对上传文件父目录的远程创建操作，则存在远程上传文件夹行为；但反过来，远程上传文件夹行为中在远程创建文件夹之后，并非一定有对该目录下文件的远程上传行为。故该算法暂且先将分析结果序列中远程创建文件夹行为识别为远程上传文件夹行为，并将与远程上传文件夹行为相关的一个或多个远程创建文件夹和远程上传文件行为整合为一，待遍历一遍分析结果序列后再进行新一轮的遍历，将上传文件列表为空的远程上传文件夹行为重新恢复为远程创建文件夹行为，完整的分析过程如图 3 所示。

远程下载文件夹行为识别算法：该算法的流程与远程上传文件夹行为的识别类似，不同之处在于上传文件夹命令执行时会先后发生远程创建文件夹和远程上传文件行为，下载文件夹则对应于远程浏览目录/搜索文件和远程下载文件行为。这里，若下载目标文件夹中全是文件而不存在文件夹，则在第一步中本文的算法 1 会将相应的遍历文件行为判定为浏览目录，否则便判定为搜索文件。

2.4 行为识别模块

经过关联分析模块的初步判定分析，异常文件管理行为全被识别，但是分析结果中可能包含部分正常网络应用程序的误报情况。正常网络应用程序在进行网络交互的同时，可能存在浏览、搜索、读写文件的操作，故有可能存在正常文件行为的误判。对此，行为识别模块通过白名单机制来排除误判的浏览/搜索行为，根据远程上传/下载文件和远程浏览/搜索文件的关系来识别真正的异常行为。

通过统计分析，正常网络应用程序浏览/搜索的目标文件集中在 AppData 目录或其程序安装/数据目录，据此制定白名单库以有效剔除误判的浏览/搜索行为。

再者，远程上传/下载与远程浏览/搜索行为存在依存

关系，即远程上传文件（夹）行为发生前的最近浏览/搜索行为一定存在，且是对上传目标文件（夹）父目录的远程浏览行为；而远程下载文件（夹）行为发生前的最近浏览/搜索行为一定存在，并且是对下载文件父目录的浏览或对其祖先目录的远程搜索行为。因而在保证远程浏览目录/搜索文件行为识别准确的前提下，通过关联分析远程浏览目录/搜索文件行为，在不影响异常行为识别准确率的同时有效地降低了远程上传/下载文件（夹）的误报率。

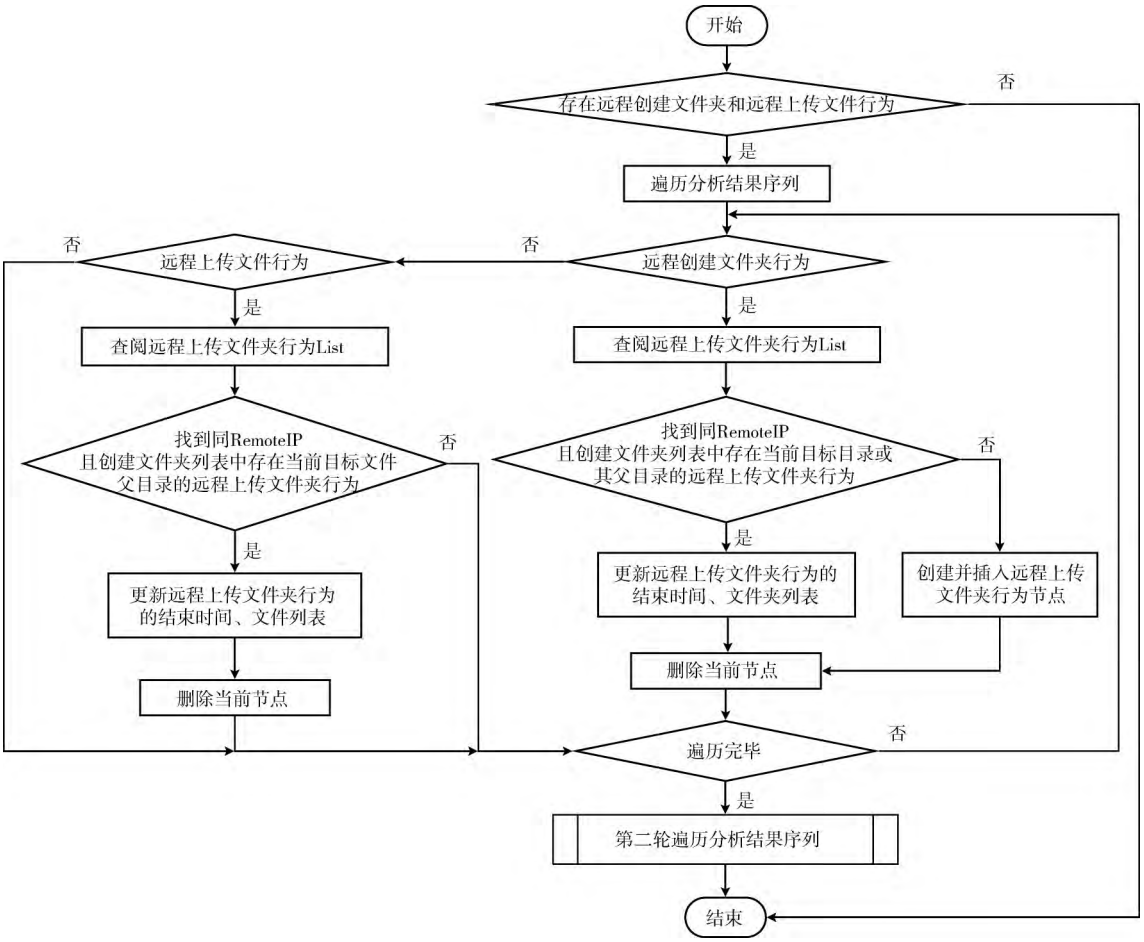

图 3 远程上传文件夹行为识别算法流程

3 实验结果与分析

本文选取了 12 个远程控制型木马样本和 45 个正常网络应用程序，使用异常文件管理活动识别系统 F-Sensor 来对异常和正常的文件管理行为进行识别，计算行为识别的检出率和误报率。

对异常文件管理行为的测试：分别对 12 个木马样本的每个文件管理行为随机触发至少 20 次，对比测试记录和识别结果并计算行为识别的检出率。表 2 列出了对异常文件管理行为的识别结果，表中的符号“—”表示木马不具备此项功能。由此可知，本系统对 6 种异常文件管理行为识别的检出率均很高。

对表 2 中未识别的异常文件管理行为进行分析并发现：

（1）对于贝壳木马，因某一测试用例中上传和下载行为的目标文件内容存在大量的重复数据，贝壳样本对该文件的压缩率高达 90%，而非本文中远程上传/下载行为关联分析算法中的 50%，导致对该文件的下载/上传行为被识别系统漏掉（上传/下载文件夹行为的漏报也是同样的原因导致），这种漏报可通过调整算法中读取/写入与发送/接收数据长度的比值得以解决；

（2）对于 PcShare 木马，因某一测试用例中搜索目标目录为单层目录，故搜索行为被识别为浏览行为，这一误判无法避免；

（3）对于 IM 木马，在浏览个别系统目录时，被控端程序同时访问并读取该目标目录下的 desktop. ini 文件，致使浏览行为被误判为上传文件夹行为；

（4）对于 Radmin 木马，若浏览目录与上传文件（夹）行为时间间隔很短，两者将被整合为对浏览目标目录的上传行为，可通过进一步调整行为序列中函数调用时间间隔得以解决。

表 2 异常行为识别结果统计

木马名称	各类型行为识别的准确率/%					
	远程浏览目录	远程搜索文件	远程上传文件	远程下载文件	远程上传文件夹	远程下载文件夹
贝壳	100	- -	95.3	91.7	95	92
灰鸽子	100	- -	100	100	100	100
黑洞	100	- -	100	100	100	100
PcShare	100	95.83	100	100	- -	100
Poison Ivy	100	100	- -	- -	- -	--
暗组	100	- -	100	100	100	100
大白鲨	100	- -	100	100	- -	--
华中帝国	100	- -	100	100	- -	--
IM	66.7	- -	100	100	- -	--
Bifrost	100	100	100	100	- -	100
Radmin	100	- -	100	100	91	100
远控 7 号	100	- -	100	100	- -	--
总计	97.26	98.61	99.57	99.26	97.2	98.86

注：PcShare 和 Poison Ivy 的搜索关键词获取不到，Bifrost 的搜索关键词大部分可获取到。

此外，有些木马（如 PcShare 和 Poison Ivy）在搜索文件时，将调用遍历行为相关 API 函数获取到的文件名与搜索关键字进行匹配，从而实现对包含关键词文件的搜寻，故搜索关键词不能反映在 API 函数的参数中。所以，PcShare 和 Poison Ivy 的搜索关键词无法通过 API Hook 的方法获取到。

对正常文件管理行为的测试：使用 F-Sensor 对 45 个正常网络应用程序进行持续 72 小时的监控，同时不定期地在测试程序的“另存为”等文件浏览窗口中浏览文件，使用测试程序上传或下载文件（夹）等，最终观察识别结果并计算行为识别的误报率。测试发现，有一个进程出现一次远程浏览目录行为的误报，其它 5 种行为不存在误报情况。因正常软件行为发生次数无法确定，故其误报率以进程数目为基数，计算得误报率为 2.22%。

4 结束语

外部入侵是导致机密或隐私文件被窃取的重要原因，而当前的软件行为检测和文件监控技术均难以自动化识别异常文件管理活动。为此，本文提出一种基于文件和网络关联分析的异常文件管理活动识别方法，能够有效识别系统中已发生的异常文件管理活动，还能以此为突破口实现对恶意软件的检测。实验结果表明，原型系统 F-Sensor 能够有效地识别异常文件管理活动，且误报率较低。利用本方法，不仅能检测恶意软件还能够重现攻击者在远程控制过程中的文件行为轨迹，为分析攻击者的控制意图提供了极大的帮助。下步工作将进一步完善分析算法，并研究针对文件名和内容的搜索关键词的获取方法。

参考文献：

[1] LI Daquan, LUO Kelu. File operations recognition based on filter driver [J]. Journal of Computer Applications, 2010, 30 (A01): 182-184 (in Chinese). [李大权，罗克露. 驱动层文件监控操作类型识别模型 [J]. 计算机应用，2010，30 (A01)：182-184.]

[2] FAN Xuebin, PANG Jianmin, ZHANG Yichi, et al. File monitoring model based on IRP feature sequence [J]. Journal of Information Engineering University, 2012, 13 (4): 508-512 (in Chinese). [范学斌，庞建民，张一弛，等. 基于 IRP 特征序列的文件行为监控模型 [J]. 信息工程大学学报，2012，13 (4)：508-512.]

[3] HAN Lansheng, GAO Kunlun, ZHAO Baohua, et al. Behavior detection of malware based on combination of API function and its parameters [J]. Computer Application and Research, 2013, 30 (11): 3407-3410 (in Chinese). [韩兰胜，高昆仑，赵保华，等. 基于 API 函数及其参数相结合的恶意软件行为检测 [J]. 计算机应用研究，2013，30 (11)：3407-3410.]

[4] LI Huanzhou, TANG Zhangguo, ZHONG Mingquan, et al. The research and implementation of Trojan detection system based on behaviors monitoring [J]. Journal of Sichuan Normal University (Natural Science), 2009, 32 (3): 386-389 (in Chinese). [李焕洲，唐彰国，钟明全，等. 基于行为监控的木马检测系统研究及实现 [J]. 四川师范大学学报（自然科学版），2009，32 (3)：386-389.]

（下转第 3182 页）

从表 1 可知，在仅获得一个异常样本的情况下，通过本文方法的不断变异，可找 EIP 被输入控制的样本，进而在此基础上可实现利用代码的编写和漏洞的验证。实验结果表明，该方法无需深入分析程序的内部逻辑，因此具有黑盒测试的易于部署、通用性强等特点。样本变异的本质是在单字节栈溢出可能导致的众多异常中，寻找一个更利于分析的异常，将异常可利用性验证转化为与经典栈溢出类似的有相对固定的方法及步骤可循的工作，因此显著提高单字节栈溢出异常分析效率。

5 结束语

本文深入分析了单字节栈溢出异常的形成原因及异常分析的难点，提出了一种单字节异常的黑盒分析方法。该方法无需深入分析程序内部语义，实现简单，能显著提升单字节漏洞的分析与利用的效率。该方法在单字节异常的确认及利用代码生成上仍需少量人工参与，无法实现完全自动化。未来工作将借助指令追踪技术，通过 Intel Pin 工具在程序执行过程中插装相应的检测代码，实现异常类型的自动判断并定位输入控制 EIP 的字段，进而实现单字节栈溢出异常的全自动分析。

参考文献：

[1] WU Zhiyong, WANG Hongchuan, SUN Lechang, et al. Survey of fuzzing [J]. Application Research of Computers, 2010, 27 (3): 829-832 (in Chinese). [吴志勇，王红川，孙乐昌，等. Fuzzing 技术综述 [J]. 计算机应用研究，2010，27 (3)：829-832.]

[2] Nagy B. Finding Microsoft vulnerabilities by fuzzing binary files with ruby-a new fuzzing framework [C] //SyScan. http://www.syscan.org/Sg/program.html,2009.

[3] Miller C. Babysitting an army of monkeys: An analysis of fuzzing 4 products with 5 lines of Python [C] //Cansecwest, 2010.

[4] Avgerinos T, Rebert A, Cha S K, et al. Enhancing symbolic execution with veritesting [C] //Proc ICSE, 2014: 1083-1094.

[5] Miller C, Caballero J, Johnson N M, et al. Crash analysis with BitBlaze [C] //BlackHat USA, 2010.

[6] Miller C. Kim Jong il and me how to build a cyber army to attck U. S. [C] //Defcon-18, 2010.

[7] !exploitable crash analyzer [EB/OL]. http://msecdbg.codeplex.com/,2013.

[8] Soulami T. Inside windows debugging [M]. Pearson Education, 2012.

[9] Cha S K, Avgerinos T, Rebert A, et al. Unleashing mayhem on binary code [C] //IEEE Symposium on Security and Privacy, 2012: 380-394.

[10] Avgerinos T, Cha S K, Hao B L T, et al. AEG: Automatic exploit generation [C] //Proceedings of the Network and Distributed System Security Symposium, 2011: 59-66.

[11] Caselden D, Bazhanyuk A, Payer M, et al. Transformation-aware exploit generation using a HI-CFG [R]. California Univ Berkeley Dept of Electrical Engineering and Computer Science, 2013.

[12] Szekeres L, Payer M, Wei T, et al. SoK: Eternal war in memory [C] //IEEE Symposium on Security and Privacy, 2013: 48-62.

(上接第 3167 页)

[5] HAN Lansheng, FU Cai, ZOU Deqing, et al. Task-based behavior detection of illegal codes [J]. Mathematical and Computer Modelling, 2012, 55 (1): 80-86.

[6] Wang C, Pang J, Zhao R, et al. Malware detection based on suspicious behavior identification [C] //First International Workshop on Education Technology and Computer Science. IEEE, 2009: 198-202.

[7] CHEN Chao, LI Jun, KONG Deguang, et al. Model checking obfuscated binary malicious code [J]. Computer Engineering and Applications, 2008, 44 (15): 61-63 (in Chinese). [陈超，李俊，孔德光. 模型检测迷惑二进制恶意代码 [J]. 计算机工程与应用，2008，44 (15)：61-63.]

[8] ZHANG Yichi, PANG Jianmin, FAN Xuebin, et al. Program malicious behavior recognizing method based on model checking [J]. Computer Engineering, 2012, 38 (18): 107-110 (in Chinese). [张一弛，庞建民，范学斌，等. 基于模型检测的程序恶意行为识别方法 [J]. 计算机工程，2012，38 (18)：107-110.]

[9] CHEN Li, ZHANG Li, YAO Yizhan, et al. Trojans control behavior detection approach based on timing analysis [J]. Computer Science, 2013, 40 (06A): 337-339 (in Chinese). [陈利，张利，姚轶崭，等. 基于时序分析的木马控制行为识别方法 [J]. 计算机科学，2013，40 (06A)：337-339.]