

基于会话关联的软件网络通信行为分析技术

杜坤凭*, 康 绯, 舒 辉, 孙 静

(信息工程大学 数学工程与先进计算国家重点实验室, 郑州 450001)

(* 通信作者电子邮箱 hnzzdkp@163.com)

摘 要:针对软件网络通信过程,提出一种基于会话关联的逆向分析方法,该方法首先对软件产生的网络通信流量和软件执行的应用程序编程接口(API)序列分别进行会话还原,再对还原的会话进行会话关联,为软件网络行为分析中的基于网络流量的分析方法和基于执行轨迹的分析方法建立了直接映射。设计并实现了相关的会话关联系统,并在此系统上进行了函数调用链的提取,使针对软件网络通信过程的分析更快捷。

关键词:软件网络通信过程分析;网络通信流量分析;应用程序编程接口序列分析;函数调用链

中图分类号:TP393 **文献标志码:**A

Behavior analysis technology of software network communication based on session association

DU Kunping*, KANG Fei, SHU Hui, SUN Jing

(State Key Laboratory of Mathematical Engineering and Advanced Computing, Information Engineering University,
Zhengzhou Henan 450001, China)

Abstract: According to the software network communication behavior, a reverse analytical method based on session association was proposed in this paper. The method restored the network traffic communication session and Application Programming Interface (API) sequence session produced by software firstly, then associated the sessions restored. Through the association, a direct mapping was built between two kinds of software network behavior analytical methods based on execution trace analysis and network traffic analysis respectively. The prototype system was designed and completed. Based on the system, the function call list was extracted. The reverse analytical method based on session association makes the reverse analysis of software network behaviors fast and convenient.

Key words: software network communication process analysis; network traffic analysis; Application Programming Interface (API) sequence analysis; function call list

0 引言

随着网络时代的到来,软件的网络通信行为越来越普遍。从人们熟知的即时通信软件、下载工具到各种网游的客户端,代理软件,以及杀毒软件,病毒木马等都存在大量的网络通信行为。分析清楚这些网络通信行为对于理解软件的内部结构,恶意代码分析,掌握网络运行状况,制定网络安全策略等均有较大帮助。

目前,对软件网络通信过程的分析主要从两方面入手:一是对软件进行逆向分析,通过逆向分析掌握软件的组织结构、执行轨迹^[1]等信息,进而理解软件的通信行为;另一方面是从软件发送和接收的数据包入手^[2],通过对数据特征的分析,达到理解软件通信过程的目的。

逆向分析方法是从小软件自身入手,对软件使用的数据结构和相关算法进行解析,从而理解软件行为。软件逆向分析又可分为静态分析和动态分析两种。对软件较全面的分析一般采用静态分析与动态分析相结合的方法。采用逆向分析技术分析软件通信过程的典型研究可参见文献[3–4]。采用软件逆向的方法,可较详细地掌握软件的通信流程、采用的通信技术、使用的保护技术、加解密强度等与软件功能和软件安

全性息息相关的信息。但逆向分析所需逆向周期较长,逆向难度也随软件保护技术的发展而大大增加。

从软件通信的数据包入手分析软件通信行为是另一种方法。这种方法是从小软件外部入手,通过抓取网络通信流量,对流量进行统计分析实现分析软件通信行为。采用此种方法,分析者不需知道流量是从何产生的,是如何产生的,只需对流量特征进行分析,便可对软件通信进行行为判定。文献[5–6]中,便是采用流量分析的方法分析软件行为。流量分析方法常用于恶意行为检测,其优点是对已知的恶意行为检测快速准确,但对于加密通信和采用私有协议的软件,单纯的流量分析还远远不够,还需配合解密技术和协议逆向技术进行分析。

上述两种分析方法相对独立,但其本质上有着十分紧密的关系。可以认为软件内部的一系列函数调用是软件产生网络数据包的源头,而网络数据包的交互是软件函数调用引发的结果。其关系如图1所示。

本文意图为两种分析方法找到结合点。使针对软件网络通信的两种研究方法关联起来,分析者既可以从软件对外通信的数据包入手,逐步逆推产生数据包的源头;也可以从软件内部入手,逐步跟踪由软件内部函数调用引发的数据交互过

收稿日期:2013-01-24;修回日期:2013-02-27。

作者简介:杜坤凭(1989–),女,云南宣威人,硕士研究生,主要研究方向:软件逆向;康绯(1972–),女,河南周口人,副教授,硕士,主要研究方向:协议安全;舒辉(1974–),男,江苏盐城人,副教授,博士,主要研究方向:恶意代码;孙静(1985–),女,江苏连云港人,硕士,主要研究方向:网络通信。

程。基于上述目标,本文提出了基于会话还原的软件网络通信过程分析方法。该方法的主要思想是,同步收集软件动态执行的函数调用序列和软件收发数据包,分别进行会话还原,再通过建立的会话关联模型进行会话关联。

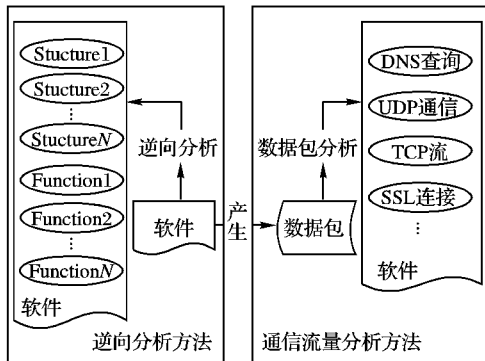


图1 两种分析方法之间关系

1 基于会话关联的软件网络通信模型

本章主要介绍了通信流量端及逆向分析端的会话还原方法,并建立了两种分析方法间的关联模型。

1.1 通信流量端的会话定义

在通信流量端,本文选择从传输层进行会话分析。因为传输层位于应用层之下,而一般应用软件的网络数据传输都位于应用层,在传输层进行会话还原,可以屏蔽不同软件使用协议的差异。传输层最重要的两大协议是传输控制协议(Transmission Control Protocol, TCP)和用户数据包协议(User Datagram Protocol, UDP)。本文选择对 TCP 和 UDP 协议进行会话还原。

定义1 TCP流是指由(源IP、源端口、目的IP、目的端口)四元组决定的TCP数据包序列,形式化描述如下:

$$TCPStream = \{ \langle P \rangle \mid \forall p, q \in P, p.protocol = TCP, p.SourceIP = q.SourceIP, p.SourcePort = q.SourcePort, p.DestinationIP = q.DestinationIP, p.DestinationPort = q.DestinationPort \}$$

定义2 UDP流是指由(源IP、源端口、目的IP、目的端口)四元组决定的UDP数据包序列,形式化描述如下:

$$UDPStream = \{ \langle P \rangle \mid \forall p, q \in P, p.protocol = UDP, p.SourceIP = q.SourceIP, p.SourcePort = q.SourcePort, p.DestinationIP = q.DestinationIP, p.DestinationPort = q.DestinationPort \}$$

定义1、2中的源端是指通信时发送首个数据包的一端。目的端对应于首个数据包中的目的端。一个TCP流或UDP流是一组目的明确的数据包序列,且流与流之间相对独立,因此,在通信流量端,本文将一个TCP流或UDP流看作一次会话过程。通信流量端的会话还原即将属于同一个TCP流或UDP流的数据包组织到一起。

1.2 逆向分析端的会话定义

软件功能的实现,是由一系列应用程序编程接口(Application Programming Interface, API)调用组成的。网络通信功能,就是由一系列网络通信API的调用实现的。本文认为,通信流量端的一次会话过程,追溯到软件内部,是由一系列网络通信API的调用引起的。

针对网络编程,Windows操作系统提供了多种编程方法,其中Winsock^[7]、WinINet^[8]和WinHttp^[9]是最常用的三个编程库。本文针对这三种编程方式进行分析。

在Winsock中,SOCKET句柄是Winsock的核心,大多数

API功能的实现都与SOCKET句柄相关,因此,在Winsock中,本文将API按照SOCKET句柄进行划分,将使用了同一个SOCKET句柄的API集合作为一个会话定义。

在WinINet和WinHttp中,使用较多的是HINTERNET句柄,因此,将WinINet和WinHttp的API按照HINTERNET句柄进行划分,将使用了相同HINTERNET句柄的API作为一个会话定义。

本文采用集合定义来对逆向分析端的会话进行描述。有:

定义3

$$P_s = \{ \langle Socket, \bigcup_{i=1}^n api_i \rangle \mid \forall i, j \in [1, n], \text{if } i < j, api_i.time < api_j.time, api_i.handle = Socket.handle \}$$

P_s 表示由SOCKET句柄划分的API序列。定义中Socket是具有如下属性的实体handle(SOCKET句柄)、SrcIP(Source IP, 源地址)、SrcPort(Source Port, 源端口)、DstIP(Destination IP, 目标地址)、DstPort(Destination Port, 目标端口)、sData(Send Data, 发送数据)、rData(Recv Data, 接收数据)、time(建立时间)。每个api有三个属性,分别是name(API名)、time(执行时间)和handle(使用句柄)。从定义3可以看出,由SOCKET定义的通信过程的API序列是按执行时间顺序排列,且同一集合中每个api使用的SOCKET句柄都相同。

定义4

$$P_h = \{ \langle HInternet, \bigcup_{i=1}^n api_i \rangle \mid \forall i, j \in [1, n], \text{if } i < j, api_i.time < api_j.time, api_i.hInternet = HInternet.hInternet \}$$

P_h 表示由HINTERNET句柄划分的API序列, HInternet具有属性hInternet(HINTERNET句柄)、api有三个属性: name、time和hInternet。

1.3 会话关联模型

通过对大量软件网络通信过程的研究发现,由定义3划分的API序列,可以直接与至少一个TCP或UDP流对应。而由定义4划分的API序列不能直接与TCP流和UDP流对应,究其原因,一方面是WinINet和WinHttp中的HINTERNET句柄在逻辑上还可以分为三层,会话层(Session)、连接层(Connect)和请求层(Request),它们之间的关系如图2所示。

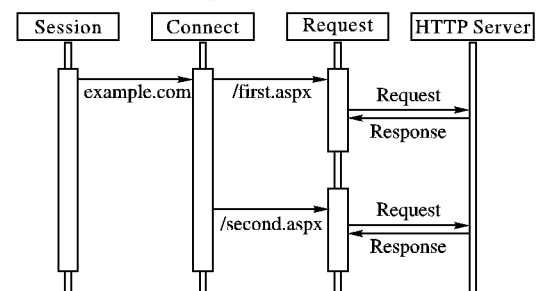


图2 WinINet、WinHttp中三层句柄关系图

通过实验发现,使用Session句柄的API只用于启动编程接口,不会产生实际的网络流量;使用Connect句柄的API用于设定连接的服务器的名称、端口、用户名和用户密码等网络通信相关属性,也不产生实际的网络流量;只有使用Request句柄的API才会引发网络通信流量。另一方面,WinINet和WinHttp提供的是针对应用层的编程方式,其操作的数据经过系统封装之后才能转换为传输层的TCP流。针对WinINet和WinHttp的上述特点,本文选择从Connect层入手,进行会话关联,也就是从使用了Connect句柄和基于Connect句柄的API序列中提取关联信息。基于定义4本文

作了如下定义。

定义 5

$$P_c = \{ \langle Connect, \bigcup_{i=1}^n api_i \rangle \mid \forall i, j \in [1, n], \text{if } i < j, api_i.time < api_j.time, api_i.hInternet = Connect.hInternet \text{ 或 } api_i.hInternet \mathbb{R} Connect.hInternet \}$$

P_c 表示通过 Connect 句柄关联的 API 序列集合。定义中的 \mathbb{R} 表示具有直接依赖关系,如图 2 所示的 Session 和 Connect, Connect 和 Request 都具有直接依赖关系。定义中的 Connect 是具有下列属性的实体: hInternet (INTERNET 句柄)、ServerName (服务器名称)、ServerPort (服务器端口)、SendData (发送数据)、RecvData (接收数据)、time (开始时间)。该集合里的 api 有三个属性, name, time 和 hInternet。属于同一个集合的 API, 要么使用的 hInternet 句柄与 Connect 的句柄相等, 要么使用的句柄与 Connect 的句柄具有直接依赖关系。

除此之外, 在 Winsock 中, 还存在两个特殊的函数, gethostbyaddr 和 gethostbyname, 这两个函数的调用, 实质上进行了一次 DNS 查询, 与数据流量端的一个 UDP 流对应。基于此, 本文设计了如下三种关联方式:

定义 6

$P_s \rightarrow \langle S \rangle$ 当且仅当 $S \in \{TCPStream, UDPStream\}$ 且有 $S.DstIp = P_s.DstIp, S.DstPort = P_s.DstPort$

$S.SendData = P_s.SendData, S.RecvData = P_s.RecvData$

$P_s \rightarrow \langle S \rangle$ 是指由 P_s 中 API 的执行引发了数据流 $\langle S \rangle$, P_s 和 $\langle S \rangle$ 要满足连接的地址和端口相等, 并且发送和接收的数据也相等。这种关联方式适合于定义 3 划分的 API 序列。

定义 7

$P_c \rightarrow \langle S \rangle$ 当且仅当 $S \in \{TCPStream\}$ 且有

$S.DstIp \approx P_c.ServerName, S.DstPort = P_c.ServerPort$

$P_c.SendData \in S.SendData, P_c.RecvData \in S.RecvData$

$P_c \rightarrow \langle S \rangle$ 是指由 P_c 中 API 的执行引发了数据流 $\langle S \rangle$, P_c 和 $\langle S \rangle$ 满足, 数据流 S 是 TCP 流, 且 P_c 连接的服务器名称经解析后与 S 的目的地址相等, 且 P_c 中的发送和接收的数据都包含在数据流 S 中。

定义 8

$P_c \rightarrow \langle S \rangle$ 当且仅当 $S \in \{UDPStream\}$ 且有

$S.DstIp \in \{dnsIP\}, S.DstPort = 53,$

$P_c.SendData \in S.SendData$

其中, P_c 表示 gethostbyname 和 gethostbyaddr 函数。 $P_c \rightarrow \langle S \rangle$ 是指由 P_c 中函数调用引发了数据流 $\langle S \rangle$, P_c 和 $\langle S \rangle$ 满足 S 是 UDP 流, S 的目的地址是本地 DNS 服务器中的 IP 地址, 端口是 53 端口, 且 P_c 发送的数据包包含在 S 的发送数据中。

2 基于的软件通信过程探测系统实现

本章介绍基于会话关联的软件网络通信过程分析 (Software Network Communication Analysis based on Session Restoration, SNCA_SR) 原型系统的设计与实现。基本思路是: 使用动态二进制平台^[10] Pin^[11] 对软件进行监控, 获取软件执行的 API 及参数信息, 并同步截取软件运行时产生的数据包, 再根据上文中的定义, 提取数据流量端和逆向分析端的通信会话, 最后, 使用上文提出的三种关联方法进行两端会话的关联。系统总体框架如图 3 所示。

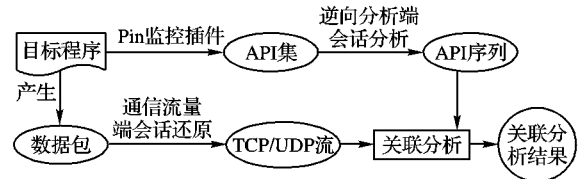


图3 基于API序列的软件网络通信关联分析系统框架

2.1 API 执行序列提取方案

定义 9 TRACE 是指具有单一入口点和多个返回点的指令序列。

定义 10 TRACE 插桩 (TRACE Instrument) 是指以 TRACE 为基本单元进行的动态二进制插桩, 分 TRACE 头插桩和 TRACE 尾插桩。TRACE 头插桩, 在 TRACE 的入口第一条指令处进行的插桩。

定义 11 RTN 是 Routine 的缩写, 是指由面向过程语言的编译器产生的函数、例程、过程。

定义 12 RTN 插桩 (RTN Instrument), 以 RTN 为基本单元进行的动态二进制插桩, 分为 RTN 头插桩和 RTN 尾插桩。RTN 头插桩是在 RTN 入口的第一条指令处进行插桩, 而 RTN 尾插桩则是在 RTN 尾部进行插桩, 根据 Pin 官方发表的帮助文档, RTN 的尾插桩是不可靠的。

API 调用信息的提取是本文研究的基础。动态二进制平台 Pin 提供了多种层次的插桩方案, 如指令插桩、基本块插桩、TRACE 插桩、RTN 插桩等。选择合适的插桩方法将直接影响到目标程序的运行效率。一般来说, 选择粒度越大的插桩方案, 对程序运行效率影响越小。要获取 API 调用信息, 理论上只需对 RTN 进行插桩, 通过 RTN 头插桩获取 API 调用信息及输入参数, 通过 RTN 尾插桩获取 API 输出参数及返回值, 但由于 RTN 尾插桩的不可靠性, API 输出参数及返回值的获取不采用 RTN 尾插桩来获取。每个 API 执行后, 都会返回到调用该 API 的指令的下一条指令处, 而从下一条指令开始时, 可以看作是一个 TRACE 的开始, 因此, 本文采用 TRACE 头插桩来判断是否为 API 的返回。整个 API 及参数获取过程如下: 在 RTN 头部记录 API 调用信息及输入参数, 并记录 API 的返回地址; 在所有 TRACE 头部插桩, 如果当前指令地址等于 API 的返回地址, 则说明该 TRACE 是 API 执行后紧接着的指令, 可在此处记录 API 的输出参数和返回值。插桩原理如图 4 所示。

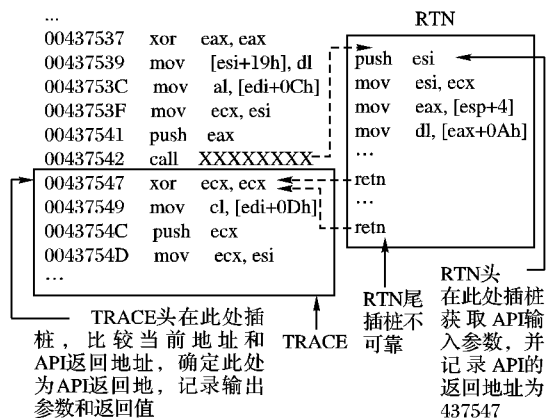


图4 API 参数提取原理

此外, 对于 API 参数的提取, 本文参考了文献[12]中设计的 API 信息库, 结合上文的 RTN 插桩和 TRACE 插桩, 实现了基于 Pin 的 API 调用序列监控插件。

2.2 逆向分析端会话提取方案设计

定义3中的Socket实体和定义5中Hinternet实体的属性,都是从API中提取而出的。本文设计了如下的描述符来描述Winsock、WinINet和WinHttp中的函数。此处以Winsock中函数为例说明,WinINet和WinHttp中描述方法与之类似。

表1 API信息描述符

描述符	说明
s;d	第d个参数标识socket句柄,d为0指返回值为socket句柄
Lip;d	第d个参数标识本地IP及端口
Rip;d	第d个参数标识远程IP及端口
sData;d	第d个参数标识发送数据
rData;d	第d个参数标识接收数据

通过表1的描述符,可为Winsock中所有API建立描述信息,表2例举了Winsock中几个API的描述信息。

表2 API描述信息示例

API	描述信息	API	描述信息
socket	s;0;	send	s;1;sData;2;
bind	s;1;Lip;2;	ntohs	NULL

通过为每个API建立描述信息,可以通过程序统一处理网络相关的API信息,快速提取网络通信端的通信会话。

通过API关联后输出如下通信会话描述结构:

Type	SrcIP	SrcPort	DstIP	DstPort	SendLen	Recvlen	SendData	RecvData	ApiList
------	-------	---------	-------	---------	---------	---------	----------	----------	---------

其中Type表示句柄类型,SOCKET句柄设为1,Hinternet句柄设为2,其余各项分别表示源IP、源端口、目的IP、目的端口、发送数据长度、接收数据长度、发送数据、接收数据、API序列。当该结构表示由Hinternet句柄关联的函数时,将ServerName和ServerPort分别存入DstIP和DstPort域。

对于gethostbyname和gethostbyaddr两个特殊函数,将Type设置为3,本地IP和端口都填0,目标IP填为本机的DNS服务器地址,端口填53,发送数据长度填待查询名字的长度,接收数据长度填0,发送数据填待查询名字。

2.3 通信流量端会话还原及关联分析

对于网络数据包,根据定义1、2以及TCP/IP协议的规范,对数据包进行会话还原,可以得到如下的TCP/UDP流描述结构:

Type	SrcIP	SrcPort	DstIP	DstPort	SendLen	Recvlen	SendData	RecvData
------	-------	---------	-------	---------	---------	---------	----------	----------

其中Type标识是TCP流还是UDP流,Type为1表示TCP流,Type为2表示UDP流。其余各项与定义1、2一致。

得到上述结构后根据本文提出的关联方法进行关联,基本过程如下:对于由SOCKET句柄定义的会话,可能产生TCP流或UDP流,要遍历所有的流进行关联。由于SOCKET编程直接运用了TCP协议和UDP协议,在协议体系中,与通信流量端还原的TCP流、UDP流都处于传输层,也就是从SOCKET相关API中提取的数据与从网络数据包中提取的数据能一一对应。所以在两端会话关联时,需保证两端所发送和接受的数据完全相等。由HINTERNET句柄关联的会话,在协议体系中位于应用层,在传输层之上,也就是HINTERNET句柄相关的API中的数据,要经过系统协议栈的再次封装才能形成网络流量端的TCP流和UDP流,所以在关联时,对于发送和接受的数据,只能采用数据查找的方式处理。也就是在所有流中查找,在目标地址与目标端口相匹配的前提下,在对应数据流的发送数据和接收数据中查找HINTERNET句柄相关的

数据,若能找到,则标记关联成功。gethostbyname和gethostbyaddr产生的会话,到UDP流中查找,如果UDP流的目标地址是本机的DNS地址,使用了53端口(DNS默认端口),并且gethostbyname/gethostbyaddr发送的数据在该UDP流中能找到,则认为该UDP流是由gethostbyname/gethostbyaddr引发的,标记为关联成功。整个会话关联算法的过程如算法1所示。

算法1 会话关联算法。

输入:通信流量端会话集<Sset>,逆向分析端按句柄关联后的API序列集<Hset>。

输出:关联信息。

算法描述:

```

For(i=0; i<Hset.size; i++)
    H=Hset[i];
    Switch(H.type)
    {
        Case 1:                                     //Socket 句柄
            For(j=0; j<Sset.size; j++)             //遍历会话流
                S=Sset[j];
                If (H.DstIP==S.DstIP && H.DstPort==S.DstPort &&
                    H.SendData==S.SendData &&
                    H.RecvData==S.RecvData)
                    RecordFindSucc();               //标记关联成功
            EndIf
        EndFor
        Case 2:                                     //HINTERNET 句柄
                                                    //根据 ServerName 获取 ServerIP
            ServerIP=GetAddrbyname(H.ServerName);
            For(j=0; j<Sset.size; j++)             //遍历会话流
                S=Sset[j];
                If (ServerIP==S.DstIP && H.ServerPort==S.DstPort
                    &&
                    S.SendData.Find(H.SendData)!=NULL &&
                    S.RecvData.Find(H.RecvData)!=NULL)
                    RecordFindSucc();               //标记关联成功
            EndIf
        EndFor
        Case 3:                                     //Gethostbyname, Gethostbyaddr
            For(j=0; j<Sset.size; j++)             //遍历会话流
                S=Sset[j];
                If (S.type==TCP)                   //只匹配 UDP 流
                    continue;
                EndIf
                If (IsDNSAddr(S.DstIP) && S.DstPort==53 &&
                    S.SendData.Find(H.SendData)!=NULL)
                    RecordFindSucc();               //标记关联成功
                EndIf
            EndFor
    }
    EndFor

```

通过上述的会话关联算法,可将软件网络通信过程中执行的API信息与其产生的网络数据包进行准确关联。会话关联后,能很好地辅助分析者对软件网络通信行为进行有效分析。对于恶意程序,一般采用指令级或函数级的行为分析方法^[13]进行分析,通过会话关联后,可以为恶意程序的相关行为找到对应的网络数据包,找到对应数据包后,可对其数据包进行针对性的特征提取,进而可形成有效的恶意程序检测方案和控制方案^[14]。对于加密通信的软件,仅从网络流量进行分析较困难,通过会话关联后,可以为加密通信的数据找到数

据发送点和数据接收点,在数据发送点逆推数据的处理流程,可提取软件的加密算法,在数据接收点跟踪软件执行流程,可提取软件的解密算法^[15]。

2.4 系统测试

本文选取了大量软件对系统进行了测试,测试结果表明本系统能将网络数据流与软件执行的 API 序列准确关联。

首先,针对文中提出的句柄关联方法,系统能准确将属于同一句柄的 API 关联,图 5 是对 QQ2011 的测试结果,图中是句柄为 1120 的 API 序列信息。

```
Socket Handle: 1120 TCP Stream
Local IP:0.0.0.0 Local Port:0
dstip:port 112.90.86.87: 80
TotalSendLen: 387 TotalRecvLen: 220
ApiList:
api:socket
api:WSAAsyncSelect
api:bind
api:connect
api:WSAAsyncSelect
api:send
data: 47 45 54 20 2f 71 6c 6f 67 6f 5f 6c 69 73 74 2e 63
Packets: 2378
api:recv
data: 48 54 54 50 2f 31 2e 31 20 32 30 30 20 4f 4b 0d 0a
Packets: 2381
api:closesocket
```

图 5 QQ2011 中 SOCKET 句柄关联示例

其中 send 函数发送的数据包与截包数据中的第 2378 号包对应,recv 函数接收的数据包与所截数据包中的第 2381 号数据包对应(截图中的数据不显示不全),通过实际对比,它们的数据的确一致。图 6 是第 2381 号包的截图。

```
# Frame 2381: 274 bytes on wire (2192 bits), 274 bytes
# Ethernet II, Src: Hangzhou_df:b8:9e (00:0f:e2:d1:b8)
# Internet Protocol, Src: 112.90.86.87 (112.90.86.87)
# Transmission Control Protocol, Src Port: http (80)
0030 19 20 8a f4 00 00
0040
0050
0060
0070
0080
```

图 6 QQ2011 运行时所截第 2381 号数据包

同样,通过 HINTERNET 句柄关联的会话,系统也能准确地将其与网络数据包关联。下面是一个 WinHttp 的测试程序,测试程序使用 WinHttp 向某 Web 服务器发送请求。

图 7 中所示的是通过值为 2439632 的 Connect 句柄关联的 API 序列图,前三行是此 Connect 句柄直接依赖的 Session 句柄信息。其中 WinHttpAddRequestHeaders、WinHttpSendRequest 函数产生了数据流量,并分别与第 21 号和 22 号数据包对应,而 WinHttpReadData 接收了数据包,所接收数据与第 26 号包对应。截图中 WinHttpSendRequest 的数据量较少,已全部显示在截图中,与之对应的第 22 号数据包的信息如图 8 所示。

```
WinHttp: test
1 WinHttpSession: 2158064
2 UserAgent: Http Post by zhaochun@163.com
3 ProxyName: Http Post by zhaochun@163.com
4 ConnectSessionInfo:
5 ConnectHandle: 2439632 UsedSessionHandle: 2158064
6 ServerName: www.easy-creator.net ServerPort: 80
7 Request Info:
8 RequestHandle: 2440288 UsedConnectHandle: 2439632
9 Header Entry Length: 70 Recv Data Length: 5042
10 ApiList:
11 api:WinHttpOpenRequest
12 api:WinHttpAddRequestHeaders
13 data: 41 6f 6e 74 65 6e 74 2d 74 79 70 65 3a 20 61 70 70 69 65 61 74 69 6e
14 Packets: 21
15 api:WinHttpSendRequest
16 data: 76 61 6c 75 65 31 3d 31 30 26 76 61 6c 75 65 32 3d 31 34
17 Packets: 22
18 api:WinHttpReceiveResponse
19 api:WinHttpQueryHeaders
20 api:WinHttpReadData
21 data: 3e 21 44 4f 43 54 59 50 45 20 68 74 6d 6c 20 50 55 4c 4e 49 43 20 22 2f
22 Packets: 26
```

图 7 对 WinHttp 中三层句柄关联示例

经测试,本系统能准确提取 API 序列进行关联,但对于网

络状况异常的通信,本系统存在一定的错误率。主要是对于网络中完全一样的两个数据流,本系统在关联时会把 API 序列关联到第一个流上。分析其原因,是在关联时,未对已关联的数据流做标记,通过修正后,该错误已不存在。

```
# Frame 22: 79 bytes on wire (632 bits), 75 bytes captured (384 bits)
# Ethernet II, Src: Hangzhou_df:b8:9e (00:0f:e2:d1:b8), Dst: Hangzhou_df:b8:9e (00:0f:e2:d1:b8)
# Internet Protocol, Src: 112.90.86.87 (112.90.86.87), Dst: 112.90.86.87 (112.90.86.87)
# Transmission Control Protocol, Src Port: http (80), Dst Port: http (80)
# Hypertext Transfer Protocol
0000 2e 6e 65 74 6d 6c 20 50 55 4c 4e 49 43 20 22 2f
0010
0020
0030
0040
```

图 8 WinHttp 测试用例所产生的第 22 号数据包

3 扩展研究

进行会话关联的主要目的是为了对软件网络行为的分析更连贯。例如分析 QQ 软件时,如果分析者想知道第 2378 号包是如何产生的,数据包中的数据都经过哪些处理,通过会话关联,分析者为网络数据包找到了数据的发送点,但对于软件分析者而言,这还远远不够。进而,本文对软件逆向端作了函数调用链提取。

函数调用链是指软件在运行到相关函数之前经过的上层函数调用序列。本文采用栈来记录调用序列。其基本思想是:当遇到函数调用时,将函数调用相关信息压栈,函数调用返回时出栈即可。这样,对任何一个函数调用,函数调用时栈的状态便是该函数的调用链。

例如,针对图 5 中值为 1120 的 SOCKET 句柄,提取的函数调用链如图 9 所示。

```
0x552ea2 call 0x563677 (1)
0x563687 call 0x56ce5e (2)
0x56ce2a call 0x46f080 (3)
0x46f858 call 0x479e90 (4)
0x479ee4 call 0x509520 (5)
0x50984f call 0x48f2f0 (6)
0x48f4a3 call 0x48f7a0 (7)
0x48f837 call socket (8) 1120
0x48fa03 call WSAAsyncSelect (8)
0x48fb2e call bind (8) 0.0.0.0
0x48fb92 call connect (8) 112.90.86.87:80
0x48fc04 call WSAAsyncSelect (8)
0x48fc2a call send (8) Packet:2378
0x48fc87 call recv (8) Packet:2381
0x48fd16 call closesocket (8)
```

图 9 值为 1120 的 SOCKET 句柄的调用链信息

通过会话关联和函数调用链的提取,软件分析人员可以从数据包入手,快速定位到数据产生和处理的相关过程,为分析软件网络通信过程提供了快捷的参考。

4 结语

本文提出了一种基于会话关联的软件网络通信过程分析方法,通过对软件运行过程中产生的数据包和软件执行的 API 序列进行会话还原和会话关联,建立了 API 序列与网络数据包序列之间的映射关系,通过建立的映射关系,可以将对 API 序列的分析结果映射到数据包序列上,对数据包序列的分析结果也可以反映出 API 序列之间的关系。在下一步研究中,将对两种分析方法间更高层次的映射作进一步的研究。

参考文献:

- [1] 康凯. 二进制程序行为检测分析平台[D]. 成都: 电子科技大学, 2010.
- [2] 应凌云, 杨轶, 冯登国, 等. 恶意软件网络协议的语法和行为语义分析方法[J]. 软件学报, 2011, 22(7): 1676-1689.
- [3] 韩纪宏. 一类互联网软件的分析与控制[D]. 上海: 上海交通大学, 2007.
- [4] 段冰. 几类互联网通信软件的分析与控制[D]. 上海: 上海交通大学, 2009.

(下转第 2066 页)

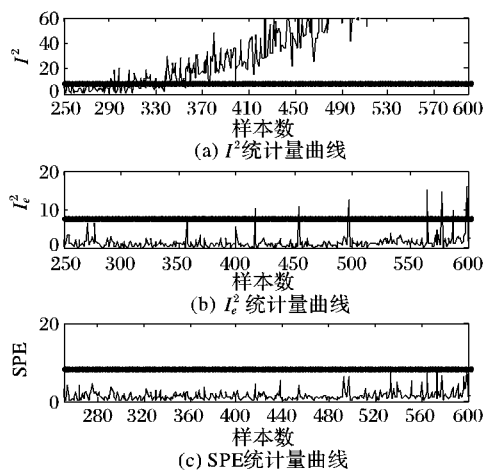


图5 子空间9 检测结果(故障2)

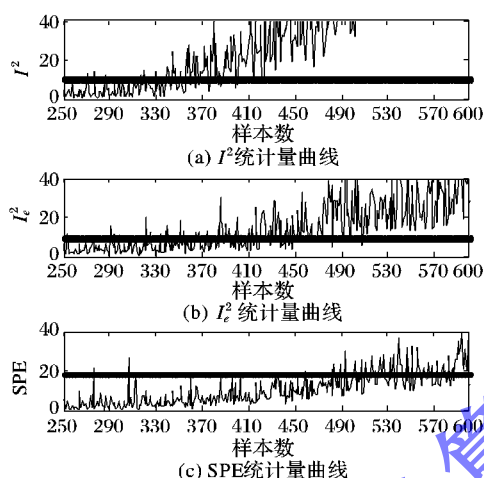


图6 仪表7 第250点模拟微小漂移ICA故障检测

4 结语

为诊断流程工业中多仪表的微小故障,本文通过独立元的贡献度将整体空间划分为多个独立元子空间,在各子空间内原有的故障诊断模型基础上增加了用于补偿的统计量。利用TE过程数据进行仿真实验研究,与整体故障诊断方法在相同故障信号下的对比,说明子空间算法和增加的补偿统计量提高了微小故障诊断的检测效果,增强了微小漂移故障检测的实时性。提供的可根据不同故障诊断需求而变动的故障诊断策略,使该方法更具有实用性和灵活性。

参考文献:

- [1] SIMOGLU A, GEORGIECA P, MARTIN E B, *et al.* On-line monitoring of a sugar crystallization process [J]. *Computers and Chemical Engineering*, 2005, 29(6): 1411 - 1422.
- [2] RUMANA S, SIRISH L S, UTTANDARAMAN S. A PCA based fault detection scheme for an industrial high pressure polyethylene reactor [J]. *Macromolecular Reaction Engineering*, 2008, 2(1): 12 - 30.
- [3] LIU X Q, XIE L, KRUGER U, *et al.* Statistical-based monitoring of multivariate non-Gaussian systems [J]. *AIChE Journal*, 2008, 54(9): 2379 - 2391.
- [4] 范玉刚, 李平, 宋执环. 基于特征样本的 KPCA 在故障诊断中的应用[J]. *控制与决策*, 2005, 20(12): 1415 - 1418.
- [5] 邓晓刚, 田学民. 基于免疫核主元分析的故障诊断方法[J]. *清华大学学报: 自然科学版*, 2008, 48(10): 1794 - 1798.
- [6] LEE J-M, YOO C K, LEE I-B. Statistical process monitoring with independent component analysis [J]. *Journal of Process Control*, 2004, 14(5): 467 - 485.
- [7] GE Z Q, SONG Z H. Process monitoring based on Independent Component Analysis-Principal Component Analysis (ICA-PCA) and similarity factors [J]. *Industrial and Engineering Chemistry Research*, 2007, 46(7): 2054 - 2063.
- [8] 王培良, 葛志强, 宋执环. 基于迭代多模型 ICA-SVDD 的间歇过程故障在线监测[J]. *仪器仪表学报*, 2009, 30(7): 1347 - 1352.
- [9] 薄翠梅, 乔旭, 张广明, 等. 基于 ICA-SVM 的复杂化工过程集成故障诊断方法[J]. *化工学报*, 2009, 60(9): 2259 - 2264.
- [10] 薄翠梅, 柏杨进, 杨海荣, 等. 多切面分类改进独立成分与支持向量机集成故障诊断方法[J]. *控制理论与应用*, 2012, 29(2): 229 - 234.
- [11] HYVARINEN A, OJA E. Independent component analysis: algorithms and applications [J]. *Neural Networks*, 2000, 13(4): 411 - 430.
- [12] HYVARINEN A. Fast and robust fixed-point algorithms for independent component analysis [J]. *IEEE Transactions on Neural Networks*, 1999, 10(3): 626 - 634.
- [13] 张沐光, 宋执环. 一种基于独立元贡献度的子空间故障检测方法[J]. *控制理论与应用*, 2010, 27(3): 296 - 302.
- [14] 张沐光, 宋执环. 独立元子空间算法及其在故障检测上的应用[J]. *化工学报*, 2010, 61(2): 425 - 431.
- [15] CHEN Q, WYNNE R J, GOULDING P, *et al.* The application of principal component analysis and kernel density estimation to enhance process monitoring [J]. *Control Engineering Practice*, 2000, 8(5): 531 - 543.
- [5] 肖枫涛, 王维, 刘波, 等. 一种基于进程流量行为的蠕虫检测系统[J]. *计算机工程与科学*, 2011, 33(4): 19 - 24.
- [6] ZHANG B, YANG J H, WU J P, *et al.* Diagnosing traffic anomalies using a two-phase model [J]. *Journal of Computer Science and Technology*, 2012, 27(2): 313 - 327.
- [7] Microsoft MSDN Library. Windows Sockets2 (Windows) [EB/OL]. [2012 - 10 - 26]. [http://msdn.microsoft.com/en-us/library/windows/desktop/ms740673\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/ms740673(v=vs.85).aspx).
- [8] Microsoft MSDN Library. About WinINet [EB/OL]. [2012 - 10 - 26]. [http://msdn.microsoft.com/en-us/library/aa383630\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/aa383630(v=vs.85).aspx).
- [9] Microsoft MSDN Library. About WinHTTP [EB/OL]. [2012 - 10 - 26]. [http://msdn.microsoft.com/en-us/library/aa383630\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/aa383630(v=vs.85).aspx).
- [10] NICHOLAS N. Dynamic binary analysis and instrumentation or building tools is easy [D]. Cambridge: University of Cambridge, 2004.
- [11] SION B. Pin - a dynamic binary instrumentation tool [EB/OL]. [2012 - 06 - 13]. <http://www.pintool.org/>.
- [12] 王乾, 舒辉, 李洋, 等. 基于 DynamoRIO 的恶意代码行为分析[J]. *计算机工程*, 2011, 37(18): 139 - 141.
- [13] 刘豫, 王明华, 苏璞睿, 等. 基于动态污点分析的恶意代码通信协议逆向分析方法[J]. *电子学报*, 2012, 40(4): 661 - 668.
- [14] 赵天福, 周丹平, 王康, 等. 一种基于网络行为分析的反弹式木马检测方法[C]//第26次全国计算机安全学术交流会论文集. 北京: 中国学术期刊电子出版社, 2011: 80 - 83.
- [15] 段刚. 加密与解密[M]. 3版. 北京: 电子工业出版社, 2010: 121 - 126.

(上接第2050页)