第 36 卷 第 9 期 计 算 机 工 程 2010 年 5 月
Vol.36 No.9 Computer Engineering May 2010
·开发研究与设计技术· 文章编号：1000—3428(2010)09—0268—03 文献标识码：A 中图分类号：TP393

基于动态二进制分析的网络协议逆向解析

何永君，舒 辉，熊小兵
(解放军信息工程大学信息工程学院，郑州 450002)

摘 要：研究未知网络协议逆向解析技术在网络安全应用中具有重要的意义。基于此，介绍网络协议逆向解析技术的发展现状，分析基于网络轨迹和基于数据流的 2 种主要解析方法，提出一种基于动态二进制分析技术的逆向解析方法，并选取 DynamoRIO 平台作为支撑，实现对数据流信息的记录和分析，从而解析出单条协议消息中主要的协议域。

关键词：协议逆向解析；数据流分析；动态二进制分析；协议域；DynamoRIO 平台

Network Protocol Reverse Parsing Based on Dynamic Binary Analysis

HE Yong-jun, SHU Hui, XIONG Xiao-bing
(Institute of Information Engineering, PLA Information Engineering University, Zhengzhou 450002)

【Abstract】Research on unknown network protocol reverse parsing is of great significance in many network security applications. This paper describes the existing circumstances of protocol reverse extraction technologies, and analyses two main reverse extraction methods, one based on network trace and the other one on data flow analysis. A method based on dynamic binary analysis is presented, which aims at extracting the main protocol fields of a single message by using DynamoRIO platform to implement data flow recording and analyzing.

【Key words】protocol reverse parsing; data flow analysis; dynamic binary analysis; protocol field; DynamoRIO platform

1 概述

协议逆向解析是指采用逆向分析的思想，解析出目标网络应用程序所使用的协议格式，在漏洞挖掘、网络入侵检测、网络管理、指纹生成、应用程序会话重放等网络安全方面具有十分重要的应用价值[1]。本文提出了一种基于动态二进制分析的未知网络协议逆向解析方法，通过分析记录的数据流信息解析出单条协议消息中主要的协议字段。

2 协议逆向解析技术

当前协议逆向解析方法主要有 2 种，即基于网络轨迹和基于数据流的分析方法。其中，基于网络轨迹的分析方法是以未知协议的网络数据为分析对象，通过对未知协议的网络轨迹进行分析推断出协议格式，特点是简单、易实现，缺点在于协议语义信息的缺乏从根本上限制了解析结果的准确性，代表性的研究成果有 PI[2], Discoverer[3]等；基于数据流的分析方法是通过分析网络应用程序对协议数据进行处理时的相关数据流信息来解析协议格式，是当前该方向的研究热点，主要研究成果有：Polyglot[1], AutoFormat[4], Tupni[5]以及文献[6]的研究成果等，与前者相比，该方法能够获得应用程序对协议数据进行处理的语义信息，而这些语义信息中通常包含了大量有关协议格式的信息，是对协议数据进行处理的主要信息来源，因而解析结果也更为准确，具有明显的优势。

基于上述考虑，本文提出了一种基于动态二进制分析技术的数据流分析方法，选取 DynamoRIO[7]作为一个具体的应用平台动态监控应用程序的执行过程，采用动态污点分析技术跟踪协议数据的传播和处理过程，记录相应的数据流信息，通过分析记录的数据流信息解析出单条协议消息中主要的协议字段。

3 DynamoRIO简介

DynamoRIO 是一个动态二进制分析平台，支持动态二进制分析技术，能在不影响程序执行结果的前提下，通过在程序动态执行过程中插入额外的分析代码，动态监控程序的执行过程。DynamoRIO 工作在操作系统和应用程序之间，它采用了代码缓存技术，通过将程序的代码拷贝进代码缓存的方式来对目标程序进行模拟执行[7]。在模拟执行过程中，用户可以对目标程序的二进制代码进行修改，实现指令级分析。

DynamoRIO 具有良好的扩展机制，用户可以利用 DynamoRIO 提供的接口编写各种动态二进制分析 DBA 工具，如污点数据分析工具等。

4 基于数据流分析的协议逆向解析技术

4.1 基于DynamoRIO的动态污点分析技术

在基于数据流分析的协议逆向解析工作中，涉及到的一个关键技术就是动态污点分析技术。文中的具体做法如下：利用 DynamoRIO 动态监控网络应用程序的执行过程，将程序从协议消息中读入到内存空间的每个输入字节标记为污点数据并对应于一个唯一的污点标签(即在输入消息中的偏移)。然后在程序执行过程中跟踪每个污点数据的处理过程，将以污点数据为输入的所有指令的输出结果标记为污点数据，且污点标签与输入数据的标签相同，此外还要考虑地址依赖。当一个污点操作数被用来计算即将载入的另一个值的地址时，就存在地址依赖。此时加载操作的结果不仅依赖于被加载的值本身(直接依赖)，还依赖于被加载值的内存地址。

4.2 基于程序执行轨迹的协议字段提取技术

对于未知网络协议的逆向解析，主要是解析出有关协议格式的主要信息，如分隔符、关键词、长度域、目标域、指针域等，对于无法确定语义信息的协议字段(二进制协议中比

作者简介：何永君(1982－)，女，硕士研究生，主研方向：网络与信息安全；舒 辉，副教授、博士；熊小兵，博士研究生

收稿日期：2009-11-18 **E-mail：**hexxxxxx123@tom.com

较常见)，通常将其解析为固定长度域(字段域的长度是固定的)和可变长度域(字段域的长度是可变的)。由于篇幅所限，并且指针域、固定长度域和可变长度域的解析都相对比较简单，因此本文着重介绍了分隔符、关键词、长度域和目标域的解析。

4.2.1 分隔符的解析

定义 1 分隔符是指用来标示协议字段结束的一个或连续多个已知值的字节。

解析的基本思想如下：通过查找用于消息中连续位置进行比较的字符来确定分隔符。为此，先将消息序列中的所有字节标记为污点数据，利用 DynamoRIO 动态监控程序对所有污点数据的处理过程，根据污点数据分析方法记录与每个污点数据处理相关的数据流信息，包括指令的地址、类型、操作码、源操作数、目的操作数等，对于污点数据操作数还要记录其污点标签。以记录的数据流信息为基础，提取出与污点数据相关的比较类指令，判断这些指令中与常量字符比较的污点标签是否连续，以此确定协议消息中的分隔符。解析算法如下：

```
separator_struct *Separators(dataflow_struct *DataFlow)
{
CmpInstrSet=get_cmp _instr(DataFlow);
//获取数据流中的比较指令集
ConCharacterSet=get_con_character_list(CmpInstrList);
//获取与污点数据进行比较的常量字符集
TaintTagSet=get_taint_tag(CmpInstrSet,ConCharacterSet);
//获取与常量字符比较的污点数据标签集
ConCharacter=get_first _concharacter(ConCharacterSet);
//获取第一个常量字符
while(ConCharacter!=NULL)
//循环处理与污点数据比较的每个常量字符
{
//将污点标签集中的连续标签合并成多个间隔区间，将不属于任
//何间隔区间的标签丢弃
MergedTagIntervals=traverse_and_merge(ConCharacter,TaintTag
List);
if(MergedTagIntervals!=NULL 且 ConCharacter 出现在 Merged
TagIntervals 中)//该字符常量为分隔符
SeparatorSet=add_separator(ConCharacter);
//将其加入分隔符集合
if(字符 ECharacters 总是位于 ConCharacter 前面或后面且程序又
对 ECharacters 进行了比较)
SeparatorSet=extend_separator(ConCharacter, ECharacters);
//扩展为多字节分隔符
ConCharacter=get_next _concharacter(ConCharacterSet);
}
return SeparatorSet;
}
```

4.2.2 关键词的解析

定义 2 关键词是指既出现在协议消息中，又出现在协议应用程序中的协议常量，如 HTTP 协议中的“GET”。

解析的基本思想如下：通过跟踪污点数据与非污点的常量字符(串)的比较来确定关键词。首先，对执行轨迹进行处理，提取出其中污点字符(串)与非污点的常量字符(串)进行比较且比较成功的指令集。然后，将这些指令集中涉及的常量字符(串)作为候选关键词。接着，在协议应用程序的二进制文件中对候选关键词进行扫描加以验证，因为协议的关键词通常都以某种方式硬编码在应用程序中了。最后，对找到的关键词，检查其后面的那些字符是否也出现在协议消息和协议应用程序中，若是，则将该关键词扩展为包括这些字符。解析算法如下：

```
keyword_struct *Keywords(dataflow_struct *DataFlow)
{
//获取数据流中污点字符与常量字符进行比较，且比较结果相等
//的所有指令
CmpSuccessInstrSet=get_cmp _successful_instr(DataFlow);
if(CmpSuccessInstrSet!=NULL)//存在这样的指令
//将这些指令中涉及的常量字符作为潜在关键词提取到潜在关
//键词集中
PKeywordSet=get_potential_keyword(CmpSuccessInstrSet);
PKeyword=get_first _pkeyword(PKeywordSet);
//获取第一个潜在关键词
while(PKeyword!=NULL)//验证候选关键词
{
if(PKeyword 出现在协议应用程序中)//正确识别出一个关键词
KeywordSet=add_keyword(PKeyword);//将其加入关键词集中
if(PKeyword 后的那些字符也出现在协议消息和协议应用程
序中)
KeywordSet=extend_keyword(PKeyword,EKeyCharacters);
//将该关键词扩展为包括这些字符
PKeyword=get_next _ pkeyword(PKeywordSet);
}
return KeywordSet;
}
```

4.2.3 长度域和目标域的解析

长度域用于确定目标域。在处理目标域时，通常程序会访问消息中的某些连续字节，这些访问大多是在一个循环中实现的，并且循环次数是受长度域控制的。因此，解析长度域时，首先利用 IDA 插件来完成一些静态分析工作以获知：(1)哪条比较指令表示的是循环跳出点；(2)每条指令属于的循环集(注：当存在循环嵌套时，一条指令可能属于多个循环)。然后，采用基于 DynamoRIO 的污点分析方法记录相关数据流信息，与前面类似。接着，利用数据流信息和静态分析结果提取出所有既是循环跳出点又对污点数据进行处理的比较指令，这些指令涉及的污点字节即为潜在的长度域。最后，对每个潜在长度域，检查其控制的循环中涉及的污点字节标签是否连续，据此确定目标域。若找到对应的目标域，则正确地识别出了一个长度域和对应的目标域。由于篇幅所限，这里将不再给出解析算法。

5 基于DynamoRIO的协议逆向解析实例测试

本文提出的解析方法主要用于未知网络协议格式的逆向解析，为验证其有效性，对文本协议和二进制协议都进行了测试，由于篇幅所限，在此只给出文本协议的测试实例。

对于文本协议的测试，本文选取 HTTP 协议为测试对象，这里重点以 GET 消息为例加以说明。

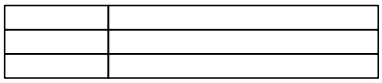
GET 消息实例：GET /index.asp HTTP/1.1\r\n

选取 IIS 作为 Web 服务器，并在客户端用 IE 浏览器提交 GET 消息，利用 DynamoRIO 动态监控 IIS 对 GET 消息的处理过程，记录相关数据流信息，最终按照 4.2 节中的解析规则逆向解析得到结果，如图 1 所示。

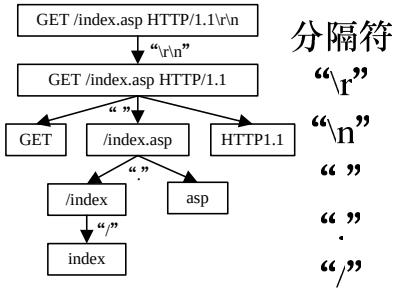
为验证逆向解析的准确性，将逆向解析结果与网络协议分析工具 Wireshark 的解析结果进行了对比(图 2)。结果表明：

逆向解析得到的结果与 Wireshark 解析得到的结果基本一致，个别不同的地方在于逆向解析中将请求的文件名细分成了文件名前缀和后缀两部分，而 Wireshark 直接将文件名的前缀和后缀看作一个整体，并包括了前面的分隔符“/”，说明逆向解析的结果基本正确。

(a)单字节分隔符的提取



(b)分隔符和关键词解析结果



(c)协议字段划分结果

图 1　HTTP GET 消息解析实例

消息序列	逆向解析结果	Wireshark 解析结果
GET /index.asp HTTP/1.1\r\n	Keyword: GET	Request Method: GET
	Field: index	Request URI: /index.asp
	Field: asp	
	Keyword: HTTP/1.1	Request Version: HTTP/1.1

图 2　GET 消息格式逆向解析结果与 Wireshark 解析结果的对比

6　结束语

本文在对当前协议逆向解析技术进行分析对比的基础上，提出了一种基于动态二进制分析的协议逆向解析方法，论述了该方法的主要思想和具体实现，并通过实例测试加以验证。测试结果表明，该方法能较好地解析出网络协议的主要格式。后续改进工作包括：细化解析粒度，以便解析出小于一个字节的协议字段；引入生物信息学中的序列比对算法，用以对同类型的多条消息进行分析，提高解析的精确度；以及实现一次可处理多条消息，将同一协议会话中的多条消息关联起来。

参考文献

[1] Caballero J, Yin Heng, Liang Zhenkai, et al. Polyglot: Automatic Extraction of Protocol Format Using Dynamic Binary Analysis[C]// Proc. of the 14th ACM Conference on Computer and Communications Security. Alexandria, USA: [s. n.], 2007.

[2] Beddoe M. The Protocol Informatics Project[EB/OL]. [2009-08-24]. http://www.4tphi.net/~awalters/PI/PI.Html.

[3] Cui Weidong, Kannan J, Wang H J. Discoverer: Automatic Protocol Reverse Engineering from Network Traces[C]//Proc. of the 16th Usenix Security Symposium. Boston, VA: USA: [s. n.], 2007.

[4] Lin Zhiqiang, Jiang Xuxian, Xu Dongyan, et al. Automatic Protocol Format Reverse Engineering Through Context-aware Monitored Execution[C]//Proc. of the 15th Symposium on Network and Distributed System Security. San Diego, California, USA: [s. n.], 2008.

[5] Cui Weidong, Peinado M, Chen K, et al. Tupni: Automatic Reverse Engineering of Input Formats[C]//Proc. of ACM Conference on Computer and Communications Security. Alexandria, VA, USA: [s. n.]. 2008.

[6] Wondracek G, Comparetti P M, Kruegel C, et al. Automatic Network Protocol Analysis[C]//Proc. of the 15th Annual Network and Distributed System Security Symposium. San Diego, California, USA: [s. n.], 2008.

[7] Bruening D L. Efficient, Transparent, and Comprehensive Runtime Code Manipulation[D]. Cambridge, USA: Massachusetts Institute of Technology, 2004.

编辑　任吉慧

（上接第 267 页）

4.2　模拟结果

模拟的最终目的是观察其动态生长过程中的形态-分布特性、变化规律等信息。在形态特征上，作为模拟结果的一个实例，图 3 给出了模拟出的荠菜根系生长过程模型。可以看出，模拟出的荠菜根系形态逼真。

(a)生长 10 天的模拟结果　　(b)生长 30 天的模拟结果

图 3　荠菜根系生长过程模拟

5　结束语

在对荠菜根系形态-分布特性及变化规律的研究中，本文基于 Visual C++平台并采用 OpenGL 图形库，依据几何模型的建模技术，提出了一种荠菜根系的三维建模方法。实践证明该方法能够准确表达荠菜根系的形态特征和生长规律，具有算法简单、效率高、实用性强等特点。在相关领域，为构建其他植物根系模拟系统提供了一种行之有效的思路和方法。

参考文献

[1] 黄雪梅，蔡　军. 荠菜的生物学特性及其开发利用[J]. 食品与药品, 2005, 7(5): 66-68

[2] 钟　南，罗锡文. 基于微分 L 系统理论的植物根系生长模拟的算法[J]. 系统模拟学报, 2006, 18(2): 138-143.

[3] 陆时万，徐祥生. 植物学[M]. 北京: 高等教育出版社, 1991.

[4] 胡包钢，赵　星. 植物生长建模与可视化——回顾与展望[J]. 自动化学报, 2001, 27(6): 816-835.

[5] Lynch J P, Nielsen K L, Davis R D. SimRoot: Modelling and Visualization of Root Systems[J]. Plant and Soil, 1997, 188(1): 139-151.

编辑　任吉慧