

恶意行为图构建与匹配算法研究

王乐乐¹ 汪斌强¹ 刘建港² 苗启广³

1 战略支援部队信息工程大学信息技术研究所 郑州 450000

2 南京信息技术研究院 南京 210000

3 西安电子科技大学计算机科学与技术学院 西安 710071

(635718080@qq.com)

摘要 恶意程序是互联网时代一个非常具有威胁性的安全问题。恶意程序的出现和传播速度的加快,使得对恶意程序的检测变得更加困难。大多数防火墙和防病毒软件都是根据恶意特征、使用一系列特殊字节来识别恶意代码。然而,恶意程序编写者会使用代码混淆技术来躲避这种检测。为此,研究者提出了动态分析方法来检测这种新的恶意程序,但这种方法的时间效率和匹配精度并不令人满意。文中提出了一种有效的恶意行为图构建与匹配算法,包括存储二维关联图的存储方法、行为图的构建方法、行为关联规则的构建方法、行为图解析算法的设计、行为匹配算法等。最后给出了实验分析,证明了该方法具有较高的检测准确率;除 Auto 类外,其对其他类别恶意程序的识别率都在 90% 以上。

关键词: 行为图;最小行为;行为关联;行为匹配

中图法分类号 TP393

Study on Malicious Behavior Graph Construction and Matching Algorithm

WANG Le-le¹, WANG Bin-qiang¹, LIU Jian-gang² and MIAO Qi-guang³

1 Institute of Information Technology, Information Engineering University, Zhengzhou 450000, China

2 Nanjing Information Technology Institute, Nanjing 210000, China

3 School of Computer Science and Technology, Xidian University, Xi'an 710071, China

Abstract Malware is a very threatening security problem in the Internet age. Due to the emergence of malicious programs and the speed up of propagation, it becomes more difficult to detect malicious programs. Most firewalls and antivirus software use a special set of bytes to identify malicious code based on malicious characteristics. However, a programmer of malicious program uses code obfuscation techniques to avoid this detection. Therefore, researchers use dynamic analysis method to combat this new malicious program, but the time efficiency and matching accuracy of this method are not satisfactory. This paper proposes an effective malicious behavior graph construction and matching algorithm, including the storage method of two-dimensional association graph, the construction method of behavior graph, the construction method of behavior association rules, the design of behavior graph parser, and the behavior matching algorithm. Finally, experimental verification analysis proves that this method has a high detection accuracy rate, except for the AutoRun category, the recognition rates for other types of malware are all above 90%.

Keywords Behavior graph, Minimum behavior, Behavior correlation, Behavior matching

1 引言

当前,黑客攻击从单打独斗逐步向体系化、规模化的 APT 攻击转变。2019 年,我国持续遭受来自“APT28”“蔓灵花”“海莲花”等 30 余个 APT 组织的网络攻击^[1]。这些攻击的意图不再是获取某一主机上的敏感信息,而是更多地向基础行业如通信、外交、金融、军工等领域延伸,大规模、有组织的 APT 攻击已经成为网络战的有力武器,使我国的网络空间安全遭受到严重威胁。因此,对海量样本程序的分析 and 检测越来越难以满足新的要求。从恶意程序的攻击过程来看,攻击者往往需要一系列的 API 操作才能完成一个复杂的网络

攻击行为。随着用户和实体行为分析(User and Entity Behavior Analytics, UEBA)^[2]概念的提出,研究人员开始从行为角度来理解恶意程序功能,重视对恶意行为本身的研究。

理解恶意程序的功能,深入挖掘恶意程序的攻击意图,进而获取恶意程序的隐藏特性,需要注意两个方面的因素:第一,全面获取恶意程序的行为;第二,精准清晰地刻画这些行为。程序运行时函数之间的调用关系,即 API 调用关系,反映了程序与操作系统之间的交互关系。由于恶意程序运行时需要通过 API 函数来与底层操作系统进行交互,因此恶意程序的行为可以通过 API 调用关系反映出来。

利用图的方法描述 API 之间的调用关系,可以建立以

API调用序列为基础的恶意行为依赖图^[3]。通过分析API序列的行为来研究恶意程序行为的依赖关系,从而在底层API函数和恶意程序行为的直观描述之间建立起联系,可以更加清楚地了解恶意程序的行为目的。

文献[4]用API调用图生成算法直接将未知程序转换成API调用图,再利用图匹配算法进行程序相似度计算。文献[5]用动态污点分析获得行为依赖图,进行未知程序的检测。文献[6]基于图结构进行恶意代码的同源性分析,准确率达到了93%。文献[7]针对Android程序特征,设计了函数调用图特征提取算法,减少了程序分析的时间,提升了分析准确率。文献[8]提取程序中的函数调用关系,用图结构进行标号和排序,并将其归一化为具有固定个数的图结构,得到了93%的检测率。文献[9]从信息论AEP概念出发,构建了API依赖图中语义相关路径提取算法,基于相关路径平均对数因子描述特征,最后用随机森林算法进行分类,达到了97.1%的检测精度。文献[10]为解决API调用图复杂化的问题,提出了一种图重分配算法,将API调用图转换为片段行为图进行表述,在保留API调用关系的基础上,消除了图中内外部的相似性行为,这种方法使得恶意程序的平均检测准确率达到95.27%。

这些用图结构进行恶意程序分析的算法均取得了较好的检测效果,但对于被分析的恶意程序,其本身行为所产生的API调用会聚合成一个或多个结构很大的图^[11]。图结构会增加算法的时空复杂度,因此在对这种高度聚合的API依赖关系图进行抽象分析时,会加大算法的复杂度,也会降低算法的灵活性。针对这一问题,本文提出“最小行为”的概念,构建行为关系依赖图,提出了基于最小行为图匹配的恶意程序分析算法(Minimum Behavior Graph Matching based Malware Classification Algorithm, MBGM-MCA),建立基于最小行为图匹配的恶意程序分析框架。

2 行为图的构建规则

恶意程序在与底层操作系统的交互过程中会对系统的各种资源进行操作,如文件、进程、网络等。从系统资源出发,将恶意程序对系统资源的包含完整语义的一次操作定义为最小行为。本文以最小行为为核心构建最小行为图,对API调用序列进行低聚合度的抽象分析,最终降低算法的分析复杂度。

2.1 行为图的描述

定义 1(行为) 行为是指具有语义的一类操作的集合^[12]。在研究恶意程序的分析问题时,本文将程序行为定义为能够实现一定功能、并对系统资源进行一系列操作的逻辑集合。这里“操作”包括修改注册表、删除文件等。

定义 2(行为图) 设 P 为结点集合, θ 为边集之间的关系, $\langle P, \theta \rangle$ 则称为行为图。

P 中的结点分为 4 类,分别是开始结点、公共结点、终止结点和提交结点。它们各自实现不同的功能,具体定义如下:

(1)开始结点(开始)。它标识了行为的开始,在匹配过程中用来识别可用行为图的存在。该结点不是API函数,它的直接后继是行为的初始API函数。

(2)公共结点(中间)。为了达到行为的目的,这个结点将

在行为图中完成某些功能。

(3)终止结点(停止)。在实现行为目的的过程中,由于某个特殊原因未能完成,导致无法实现目的行为。在此结点,行为图匹配过程将被终止。

(4)提交结点(完成)。如果该结点匹配成功,则成功完成相应的行为匹配。

θ 分为两种关系,分别是 AND(与)和 OR(或)。

定义 3(最小行为) 最小行为指程序对系统资源的包含完整语义的一次操作。

定义 4(最小行为图) 最小行为图是用图结构对程序最小行为进行描述。它由API函数组成,描述了API函数序列的完整操作和目的,最小行为图中至少包含一种实现程序行为目的的操作。

定义 5(敏感资源) 敏感资源是指操作系统中的文件、内存、注册表、网络连接、服务、进程等关键资源,恶意程序入侵系统后常常会对这些资源进行操作,从而对系统造成破坏。与操作敏感资源相关的数据统称为敏感数据^[13]。

2.2 最小行为图的建立规则

最小行为图的建立包括 3 个层次:API函数、最小行为图 and 最小行为图描述。它的形成关系是 API 函数 $\rightarrow$ 最小行为图 $\rightarrow$ 最小行为图描述。

本文用与或(And/OR)关系图^[14]来实现行为图的构建,这一方法通过分析恶意程序在运行时所调用的API函数的时序关系,获得不同抽象层次下恶意程序运行时的行为特征。建立匹配算法的前提是前序结点具有AND(与)/OR(或)关系,并且后续结点也具有AND(与)/OR(或)关系。结点关系如图1和图2所示。

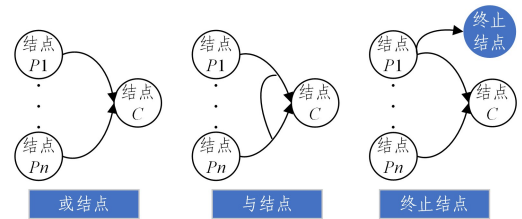


图 1 前序结点的 AND(与)/OR(或)关系
Fig. 1 AND/OR relationship of pre-order nodes

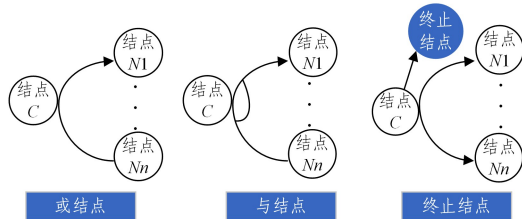


图 2 后继结点的 AND(与)/OR(或)关系
Fig. 2 AND/OR relationship of successor node

最小行为图的建立规则如下:

- (1)在最小行为图中,如果实现一个函数至少需要两个或多个子函数中的任意一个,则使用 OR(或)关系来描述。
- (2)在最小行为图中,如果实现一个函数需要两个或多个子函数,并且子函数之间存在控制依赖关系或数据依赖关系,则使用 AND(与)关系来描述。

(3)在最小行为图中,如果实现一个函数需要两个或多个子函数,并且在子函数之间没有控制依赖和数据依赖关系,则使用终止关系来描述。

根据最小行为图的建立规则,构造修改注册表项的行为图,如图 3 所示。

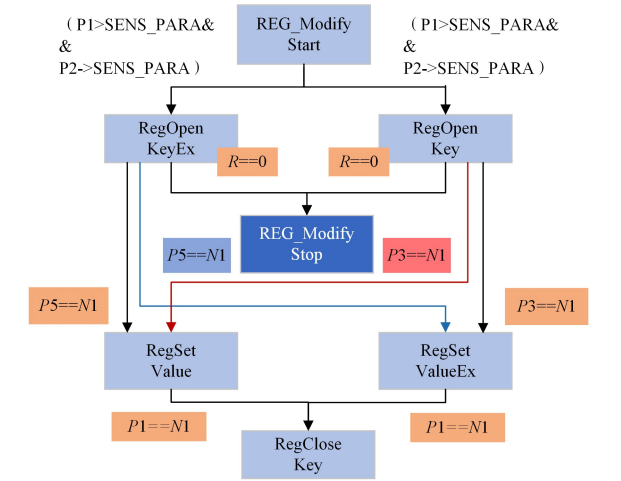


图 3 修改注册表键值的行为图

Fig. 3 Behavior diagram of modifying register key value

如图 3 所示,打开注册表、设置注册表项和关闭注册表 3 个子函数有控制依赖和数据依赖关系。只有当这 3 个操作都按顺序成功匹配时,修改注册表项的行为才会完全匹配。RegOpenkey和 RegOpenkeyEX 是两个操作之间的或关系,因此只有其中一个可以完成。从开始结点到两个结点,必须满足第一个参数和第二个参数属于敏感数据。当两个操作返回值为非零时,由于打开注册表项失败,将到达结点 REG-ModifyStop 以结束此行为的匹配过程。当 RegOpenkeyEX 和 RegsetValue 具有相同的操作句柄(P5=N1)时,可以认为已经成功地匹配了 RegsetValue。如果 ResetValueEX 与 RegClosekey 有相同的操作句柄,那么认为修改注册表项的行为已经完成。然后,报告修改注册表键值的行为匹配已经完成。

2.3 最小行为图的存储形式

恶意行为图和相关信息用关系数据库来存储,其主要存储 4 个方面的内容,包括用于匹配的 API 函数、由 API 函数组成的最小行为图、敏感数据和最小行为图的描述。相应地,在数据库中有 4 种不同结构类型的关系表来存储这 4 方面的内容。

(1)存储 API 函数的基本表

表中存储 API 函数的名称、参数及函数的描述信息,主要用于在数据导入过程中过滤掉被捕获但不需要分析的 API 函数,以减少匹配行为的时间。过滤过程的 API 函数可以应用到数据捕获过程中,由于不需要捕获 API 函数,故不需要收集函数名和参数等相关信息。存储 API 函数的基本形式的数据结构如图 4 所示。

FUNC_ID	FUNC_NAME	FUNC_PARA_NUM	FUNC_DESP
---------	-----------	---------------	-----------

图 4 基本表数据结构

Fig. 4 Data structure of basic table

FUNC_ID:在表中唯一标识一个函数或一个记录。
FUNC_NAME:用于过滤的名称。
FUNC_PARA_NUM:函数中参数的个数。
FUNC_DESP:函数的描述。

(2)存储最小行为的表

一张存储最小行为的表对应一个最小行为。在该表中,最小行为可以定义为完成某一特定功能行为,但区别于其他 API 函数组合序列所表示的行为。这一概念中包含了“最小行为”的 3 个特性:“可以定义”表示这种行为可以用一个或几个词来清楚地描述其功能和目的;“完成某一特定功能”表示这种行为实现了一个完整的功能;“区别于其他 API 函数组合序列所表示的行为”表示这种行为是为了表现出自己的特征,不仅具有公共特征,而且具有与其他行为不同的特征。这 3 个特性相互关联,共同用于表征最小行为。例如,用于修改注册表键值行为的 API 函数有:打开注册表项、设置注册表项、关闭注册表项。3 个函数代表对应注册表的 3 个操作,缺少任何一个操作都不能完整地表示修改注册表项这一行为。如果没有打开注册表项的操作,就不知道如何设置注册表项。如果没有设置注册表项的操作,则不能确定该行为是用于修改注册表项的值。在缺少关闭注册表项的情况下,无法知道注册表项的更改是否已经完成。

图 3 定义了满足 3 个属性最小行为的最小操作,完成 3 个最小操作则表示完成了修改注册表项这一行为。该行为区别于其他行为之处在于,这 3 个最小操作所代表的行为是修改注册表项,而不是创建、删除注册表项或其他操作。

为了方便匹配算法,对数据表中最小行为结构进行定义,如图 5 所示。

NODE_ID	FUNC_NAME	NODE_TYPE	PRIOR_IDS	PRIOR_REL	NEXT_NUM	NEXT_REL	NEXT1_ID	NEXT1_RULE	...	NEXTn_ID	NEXTn_RULE
---------	-----------	-----------	-----------	-----------	----------	----------	----------	------------	-----	----------	------------

图 5 最小行为表数据结构

Fig. 5 Data structure of minimal behavior table

NODE_ID:在行为图中,唯一标识一个 API 函数或一个结点。
FUNC_NAME:表示当前结点所代表的 API 函数,并且在匹配行为中使用该函数名。
NODE_TYPE:当前结点类型。本文定义了 4 种类型结点:开始结点、公共结点、提交结点、终止结点。
PRIOR_IDS:一个 ID 字符串,由行为图中当前结点的所

有直接前序结点的 ID 字符串组成。例如:对于结点 A,前序结点有 B,C,D,对应的标记为“000001”“000002”“000003”,那么 PRIOR_IDS 这个值可表示为“000001000002000003”。
PRIOR_REL:表示当前结点与直接前序结点的关系。
NEXT_NUM:表示直接后继结点的数量。
NEXT_REL:表示后继结点的相关性。
NEXT1_ID:标记第一个后继结点,可用于查找后继结点 1。

NEXT1_RULE:当前结点和后继结点的关联规则。在匹配过程中,从当前结点到结点 1 必须满足关联规则。

NEXTn_ID:标记第 n 个结点的后继结点,可用于查找后继结点 n 。

NEXTn_RULE:当前结点和后继结点 n 的关联规则。

由于每个结点的后继结点数不确定,因此 n 值也不确定。本文定义了数据库中不同行为的最大后继结点数。

(3)存储最小行为描述的表

在图 6 中,每个记录对应最小行为的描述,包括行为名称、行为权重和行为描述。如果一个最小行为匹配成功,则用表来描述并记录最小的匹配行为,报告行为匹配成功。

BEH_ID	BEH_NAME	BEH_WEIGHT	BEH_DESP
--------	----------	------------	----------

图 6 最小行为表数据结构描述

Fig. 6 Data structure description of minimal behavior table

BEH_ID:在此表中唯一标识一个行为或记录。

BEH_NAME:标记一个行为。

BEH_WEIGHT:行为的权值,从中可以确定待检测的程序是否为恶意程序。

BEH_DESP:对行为的描述,表示此行为已经执行的操作,以及此行为的目的。

(4)存储敏感数据的表

此表存储与操作敏感资源相关的敏感数据。例如,在根目录中创建文件和在系统目录下创建文件是很少见的。有些目录下的文件不应是其他文件格式,比如在字体目录“\font\”和系统“\system\”中不会有可执行文件。注册表项“Run”和“RunService”用于启动进程和服务,许多恶意程序对其进行修改以实现动态加载。如果在这些目录或文件下有操作,则可能是恶意程序的操作。本文把所有与敏感资源相关的敏感数据存储在敏感数据表中,其数据结构如图 7 所示。

DATA_ID	DATA_STR
---------	----------

图 7 敏感数据表数据结构

Fig. 7 Data structure of sensitive data table

DATA_ID:在此表中唯一标识敏感数据。

DATA_STR:将敏感数据以字符串的形式存储。

3 基于最小行为图的匹配算法

3.1 最小行为的关联规则

根据 LR 分析规则,本文设计实现了匹配最小行为的关联规则。关联规则的设计遵循词法分析和句法分析规则。在分析匹配的过程中,每个关联规则都是一个布尔表达式,最终结果是一个布尔值,根据该值可以确定是否建立了关联规则。

数据集 = { $R, C_i, N_i, \# \text{ Constant}, \$ \text{ String } \$$ }

其中, R 表示当前结点的返回值; C_i 表示当前结点的第 i 个参数; N_i 表示当前结点某个后继结点的第 i 个参数; $\# \text{ Constant}$ 表示可用于除法、乘法、加法和减法等操作的数据; $\$ \text{ String } \$$ 表示一个常量字符串。本文使用符号“ $\$$ ”作为分隔符。当字符串中有“ $\$$ ”字符时,可使用符号“ $\backslash$ ”来替换。

运算符集合 = { $+, -, \times, /, >, <, >=, <=, <>, =, ->, \&, \|, !, (,)$ }

其中, $+, -, \times, /$ 用于实现加法、减法、乘法和除法的运算。 $>, >=, <, <=, <>$ 用于实现关系运算符,“ $<>$ ”代表“不等于”,“ $=$ ”代表“相等”。“ $->$ ”运算符代表“存在运算符”,其操作的第一个参数是字符串,第二个参数是敏感数据库的名称。在数据库中查找字符串,如果数据库中存在字符串,则返回 true,否则返回 false。

3.2 基于最小行为图的匹配算法

构建行为图匹配算法前,先对要用到的结点进行定义。下面定义 9 种类型的结点。

开始结点(Start Node):行为的起点。

公共结点(Common Node):用于完成行为的某些子操作的 API 函数。

提交结点(Submit Node):当此结点成功匹配时,整个行为完全匹配成功。

终止结点(Terminate Node):当某些 API 函数无法完成其功能时,匹配过程将终止于该结点。

当前结点(Current Node):此时匹配的 API 函数。

前序结点(Prior Node):结点应出现在某个结点之前,并与该结点有关系。

后继结点(Next Node):结点应出现在某个结点之后,并与该结点有关系。

有效结点(Validate Node):API 函数已成功匹配,并且根据此结点的 NEXT_REL,下一结点匹配不成功。当一个有效结点的下一结点根据结点的 NEXT_REL 匹配成功时,有效结点将成为无效结点。

无效结点(Invalidate Node):API 函数未成功匹配或结点的下一个结点已根据 NEXT_REL 成功匹配。

行为匹配过程共分为导入数据、初始化和行为匹配 3 个步骤,具体如下。

(1)导入数据

打开要分析的文件,在基本表中查找需要分析的 API 函数,读取文件中的数据,存储到控制列表中。

(2)初始化环境

读取所有最小行为表,打开最小行为表,读取第 k 个最小行为的起始结点,写入有效结点列表中。

(3)行为匹配

行为匹配算法如算法 1 所示。

算法 1

初始化:导入数据;读取所有最小行为表;

```
1. for ( i=0 to NumOfRecList )
2. {
3. GetRecAPIName( i ); //在有效结点列表中获取第 i 个函数名称
4. for ( j=0 to NumOfValidList ) //确定当前结点的所有前序结点是否都是有效的
5. {
6.   for ( k=0 to NextNumOfValid(j) ) //确定有效结点的下一个结点是否有效
7.   {
8.     //如果当前结点的名称与有效结点的下一个结点之一的名称相同,则规则成功匹配
9.     if ( RecAPIName == NextName( k ) of Valid( j ) && RuleMatch() )
10.    {
```

```
11. if (IsIDsValid()) //确定当前结点是否有效
12. for (m=0 to NumOfPriors)
13. //确定当前结点的每个前序的所有下一个结点是否都是有效的
14. if (IsPriorNextsValid()) {
15. //如果前序结点不是起始结点,则删除先前结点
16. PriorDelete()s
17. }
19. //确定当前结点不是提交结点,终止结点不是当前结点的下一结点
20. if (!IsSubmit() && !IsNextofCurStop()) {
21. //将信息附加到验证结点列表中,并将参数附加到参数列表中
22. InsertToValid()
23. }
24. else if ( IsSubmit() ) //当前结点是提交结点
25. InsertToBeh(); //将信息放入行为列表控制框中
26. else //当前的下一个结点是终止结点
27. Nop() //Do nothing
28. }
29. }
30. }
31. }
```

3.3 关系规则的解析

为了更好地分析关系规则,对关系规则进行解析,先做下列规定:

- (1)运算器没有优先级概念,为了表达优先级,可以使用符号“()”作为分隔符,按左边优先的原则进行顺序操作。
- (2)运算模式由运算器决定。例如,对于符号“A && B”,认为“A”和“B”是布尔值。除了符号“(”之外,堆栈中只有一个数据和一个运算符。

关系规则解析算法如算法 2 所示。

算法 2

输入:关系规则 R;关系规则中的所有字符的数量 K;符号栈 Z;输入的字符 c

输出:是否匹配成功(“是”或“否”)

初始化:Operator1=“+ * = < >”&!| -” //运算符 1

Operator 2=“= | > &.” //运算符 2

```
1. for i=0 to K //处理关系规则中的所有字符
2. rch1←GetRuleStr(R,i); //获取规则字符串的第 i 个字符
3. if c is 字符是符号“(”
4. push(c,Z);
5. else if c ä (Operator1) // 如果字符是运算符
6. if c is “)” and empty(Z) //如果符号“(”和符号“)”不匹配,认为关系规则错误
7. showerror;
8. if c != “)”
9. rch2 ←GetRuleStr(R,i+1)
10. if rch2 ä Operator2 //运算符是双字符
11. push (rch1 + rch2,Z);
12. else
13. push (c); // 如果运算符是单个字符
14. else if empty(Z) and {c!= “)” or Z(top) is “(” and c!= “)” }
15. //如果堆栈为空或堆栈顶部的字符为符号“(”,则推送数据和运算符
16. push (c,Z); // 推送数据
```

```
17. rch2←GetRuleStr(R,i+1)
18. if rch2 ä Operator2 //运算符是双字符
19. push (rch1 + rch2,Z);
20. else
21. push (c);
22. else if empty(Z) and c= “)” //如果字符是符号“)”,则执行缩小操作
23. if Z[top-2] != “(” //如果符号“(”和符号“)”不匹配
24. showerror;
25. else // Do the operation of reduction
26. op ← pop(Z) //获取运算符
27. d1 ← pop (Z); //获取操作数 1
28. pop (Z); //The symbol “(” is pop out
29. d2 ← pop (Z); //获取操作数 2
30. output result
31. end for
```

4 基于最小行为图匹配的恶意程序分析框架

基于最小行为图匹配的恶意程序分析框架可分为 3 个模块:数据捕获模块、行为匹配模块、行为分析模块。每一个模块都有各自的输入和输出信息,在每两个模块之间设置一个接口和规则,这样 3 个模块可以独立实现各自的功能,以实现更高的匹配效率。整个分析框架如图 8 所示。

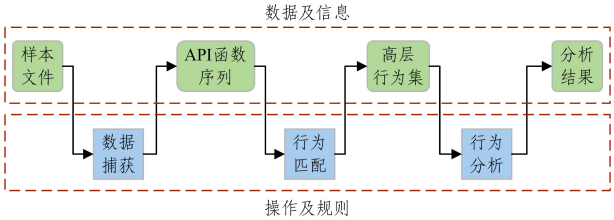


图 8 基于行为图匹配的恶意程序分析框架
Fig. 8 Malware analysis framework based on behavior graph matching

(1)数据捕获模块

数据捕获是通过 Cuckoo Sandbox^[15]来实现的,Cuckoo Sandbox 捕获程序在运行过程中的恶意行为。在此阶段,输入数据是测试样本,输出结果是 API 序列的捕获报告。

(2)行为匹配模块

行为匹配模块采用客户端/服务器的模式进行,如图 9 所示,客户端主要实现 3 个功能:

- 1)预处理模块将捕获报告导入分析环境;
- 2)与服务器交互,进行数据匹配;
- 3)依据行为之间的关联规则设计解析器。

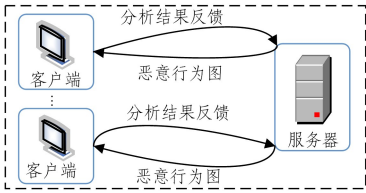


图 9 行为匹配示意图
Fig. 9 Behavior matching diagram

行为图以关联映射的形式存在于数据库中。在服务器端,可以实现行为更改、删除、添加和其他管理操作。客户端实现行为匹配,根据行为数据库中捕获的 API 函数进行匹

配,并在行为分析结束后反馈给数据库。

服务器主要管理行为图,并与客户端一起配合完成匹配过程。服务器和客户端可以配置在不同的主机上,形成分布式资源共享模型;也可以在同一主机上配置,形成集成的资源垄断模型。前者适用于网络分析环境,将数据库存储在服务器端,并在客户机上设置匹配接口,即可以实现实时数据分析。多个客户端可以同时处理,以提高分析的效率。在此阶段,输入数据是捕获的 API 序列,输出结果是敏感行为。

(3)行为分析模块

为了对敏感行为进行分析,按照最小行为的关联规则和基于最小行为图匹配算法,判断测试样本是否是一个恶意程序。可以把 2 种、3 种或者更多的敏感行为进行组合,使其恶意程度达到最大,判别其组合后的结果是否有效,直至组合结果是无效为止,最终实现测试样本的评估和恶意分类。

在此阶段,输入数据为敏感行为,输出结果为分析报告。

5 实验与结果分析

实验环境:用 Cuckoo 沙盒模拟 Windows XP 系统;样本运行时调用的 API 为 Win32 API;使用 Python 语言(Python 2.7)。

实验数据集中有 2000 个样本程序,包括 1000 个正常样本和 1000 个恶意样本。样本中有 629 个行为,通过分析这些行为,构建了 82 个行为图,如创建注册表项、修改注册表项、删除注册表项、服务创建、代码注入 Exe 文件、进程代码注入、DLL 注入、文件创建、文件搜索、文件删除等。实验数据样本为某研究所提供。

(1)恶意行为捕获能力实验

首先对最小行为图匹配算法的恶意行为捕获能力进行实验,即通过匹配算法对测试样本中的恶意行为进行捕获,并与几个有代表性的沙盒工具 NormanSandbox^[16],Anubis^[17]和 CWSandbox^[18]进行对比。

用木马程序 Trojan-Spy. Win32. Delf. enl 进行测试,用 MBGM-MCA 算法对沙盒捕获到的行为进行匹配。同时用典型的 NormanSandbox,Anubis 和 CWSandbox 3 个沙盒对该木马程序进行分析,发现其恶意行为。表 1 是用以上几种工具检测到的木马程序 Trojan-Spy. Win32. Delf. enl 所表现的行为情况,MBGM-MCA 算法可以捕获到 23 个最常见的行为。实验结果表明,MBGM-MCA 算法通过最小行为图对沙盒捕获到的行为进行了匹配,相比其他 3 个沙盒,能够捕获更多的恶意行为,具有较高的效率。

表 1 捕获到的恶意行为

Table 1 Captured malicious behaviors				
行为	MBGM-MCA	CWSandbox	Anubis	Norman Sandbox
Process_WinExec_可执行文件	✓	✓	✓	✓
File_Attributes_Get_可执行文件	✓	✓	✓	✓
File_Attributes_Get_系统关键文件夹	✓	✓	✓	✓
File_Attributes_Get_可执行文件夹	✓	✓	✓	✓
Process_Create_系统关键文件夹	✓	✓	✓	✓

(续表)

行为	MBGM-MCA	CWSandbox	Anubis	Norman Sandbox
Remote_Thread_Create_X	✓	✓	✓	✓
Thread_Priority_Get_X	✓	✓	✓	✓
DLL_Load_系统动态链接库名	✓	✓	✓	✓
Thread_resume_X	✓	✓	✓	✓
Thread_Context_Set_X	✓	✓	✓	✓
Process_Memory_Write_X	✓	✓	✓	✓
Process_Memory_Read_X	✓	✓	✓	✓
Get System Catalog	✓			
Network Information Registry Query	✓	✓	✓	
Operation-driven\Device\ip	✓		✓	
Network ICMP, DLL Loading	✓			✓
Rsaenh. dll Registry Information Query	✓		✓	
Rsaenh. dll Injection	✓			
Create a Remote thread in Rsaenh. dll	✓			
Network Port Scanning	✓	✓	✓	✓
Mutex Creation and Waiting	✓	✓		✓

(2)恶意程序识别率实验。

除了对最小行为图匹配算法的捕获能力进行实验外,本文还设计了匹配算法对于测试样本识别能力的实验,目的是检验匹配算法能否将测试样本进行正确分类,以进一步验证匹配算法的检测能力。采用的数据集和评价标准如下。

实验中的数据集包含病毒、蠕虫、木马和后门四大类 7 种共计 1000 个样本,每一种恶意程序的样本数量如表 2 所列。

表 2 7 种测试样本的数量分布

Table 2 Distribution of seven kinds of test samples			
样本名称	名称缩写	数量/个	
Virus. Win32. Downloader	Downloader	66	
Worm. Win32. Agent	Agent	98	
Worm. Win32. Delf	Delf	100	
Backdoor. Win32. Agobot	Agobot	137	
Worm. Win32. AutoRun	AutoRun	354	
Trojan-Spy. Win32. Ardamax	Ardamax	119	
Trojan. Win32. AntiAV	AntiAV	126	

评价标准:实验中使用召回率(Racall)、误检率(False Positive Rate,FPR)以及混淆矩阵(Confusion Matrix)对实验结果进行评估。

表 3 列出了在多分类任务中,MBGM-MCA 算法在 1000 个测试样本上的混淆矩阵,从中可以得到 MBGM-MCA 算法对不同类型恶意程序的混淆矩阵,并计算出算法对不同类型恶意程序的召回率和误检率,如表 4 所列。

表 3 多分类任务中 MBGM-MCA 算法在测试集上的混淆矩阵

Table 3 Confusion matrix of MBGM-MCA algorithm on test set for multi classification task							
	Downloader	Agent	Delf	Agobot	AutoRun	Ardamax	AntiAV
Downloader	60	0	0	0	4	1	1
Agent	2	89	1	1	3	1	1
Delf	1	0	95	1	2	1	0
Agobot	1	1	1	129	3	1	1
AutoRun	2	10	8	12	300	9	13
Ardamax	2	1	1	2	4	108	1
AntiAV	0	0	1	1	3	1	120

表 4 不同样本的召回率和误检率

Table 4 Recall rate and false positive rate of different samples

样本类型	召回率/%	误检率/%
Downloader	90.9	0.94
Agent	90.81	1.46
Delf	95	1.47
Agobot	94.16	2.15
AutoRun	84.75	3.06
Ardamax	90.76	1.73
AntiAV	95.24	2.13

从表 3 和表 4 可以看出,基于行为图的匹配算法对表 2 所列的 7 种恶意程序均能够有效识别,除了对 AutoRun 类的识别率较低外,对其余 6 类的识别率都达到了 90% 以上,对 Agobot, AntiAV 和 Delf 的识别率更是达到了 94% 以上。由于表 2 中对 AutoRun 类样本的分类是按照 kaspersky 给出的,与样本的真实分类有偏差,因此造成了 AutoRun 类的识别率较低。

表 3 中的实验数据也反映出部分样本产生了错分,算法存在一定的误检率。特别地,对于 AutoRun 类恶意程序的识别,误检率为 3.06%。原因是 Kaspersky 对 AutoRun 类和 AntiAV 类恶意程序的分类中所使用的 API 特征维数较多,构造的行为图较为复杂,导致其识别率低或者错误率升高。

结束语 为了直观描述恶意程序的攻击行为,获取恶意程序的真实攻击目的,本文用图结构对恶意程序进行描述和分类。在沙盒捕获的 API 序列基础上,定义了最小行为的概念,建立以“最小行为”为基础的行为图,用二维数据库对最小行为和其他参数进行存储。本文设计实现了最小行为图匹配算法,从结点和边两方面进行行为的比较和匹配,建立了基于最小行为图匹配的恶意程序分析框架,从待测样本中捕获了 629 种行为,构造了 82 个常见的最小行为图。实验表明,与常见的沙盒工具相比,采用 MBGM-MCA 算法可以捕获大部分的恶意行为,其捕获能力要高于常见的沙盒工具。除 AutoRun 类外,其对其他类别恶意程序的识别率均能达到 90% 以上。可见,本文所设计的 MBGM-MCA 算法在检测率和识别率上均能达到较高的性能。

参 考 文 献

[1] CERTNET/CC. 2019 CNCERT Cybersecurity analysis [EB/OL]. https://www.cert.org.cn/publish/main/46/2020/20-200420191144066734530/20200420191144066734530_.html.

[2] LUKASHIN A, POPOV M, BOLSHAKOV A, et al. Scalable Data Processing Approach and Anomaly Detection Method for User and Entity Behavior Analytics Platform[C]//International Symposium on Intelligent and Distributed Computing. Springer, Cham, 2019:344-349.

[3] CHENG B, TONG Q, WANG J, et al. Malware clustering using family dependency graph[J]. IEEE Access, 2019, 7: 72267-72272.

[4] ELHADI A A, MAAROF M A, BARRY B I, et al. Enhancing the detection of metamorphic malware using call graphs[J]. Computers & Security, 2014, 46(oct.): 62-78.

[5] NIKOLOPOULOS S D, POLENAKIS I. A graph-based model for malware detection and classification using system-call groups [J]. Journal of Computer Virology & Hacking Techniques, 2017, 13(1): 29-46.

[6] ZHAO B L, MENG X, HAN J, et al. Homology analysis of malware based on graph[J]. Journal on Communications, 2017, 38(Z2): 86-93.

[7] LIN S J. Research of android malware detection technology based on function call graph[D]. Beijing: Beijing University of Posts and Telecommunications, 2017.

[8] LI L. Graph Structure Oriented Android Malware Detection [D]. Beijing: Beijing Jiaotong University, 2018.

[9] ZHAO C R, ZHANG W J, FANG Y, et al. Malware detection based on semanticAPI dependency graph[J]. Journal of Sichuan University(Natural Science Edition), 2020, 57(3): 78-84.

[10] XIAO F. Research on Malware Detection Method Based on Behavior Analysis [D]. Beijing: Beijing University of Posts and Telecommunications, 2020.

[11] FREDRIKSON M, JHA S, CHRISTODORESCU M, et al. Synthesizing Near-Optimal Malware Specifications from Suspicious Behaviors[C]//IEEE Symposium on Security & Privacy. IEEE, 2010: 45-60.

[12] MIAO Q G, WANG Y, CAO Y, et al. Research on detection technology of malicious software based on sub-behavior[J]. Systems Engineering and Electronics, 2013, 34(8): 1735-1740.

[13] LIU W C. Research on Analysis Technology of Malware based on Minimum-Behavior[D]. Xi'an: Xidian University, 2012.

[14] MARTIGNONI L, STINSON E, FREDRIKSON M, et al. A Layered Architecture for Detecting Malicious Behaviors[C]//International Symposium on Recent Advances in Intrusion Detection(RAID 2008). Springer-Verlag, 2008.

[15] Cuckoo[EB/OL]. <https://cuckoosandbox.org/>.

[16] BAI J, SHI Q. Malware Detection Method based on Dynamic Variable Length API Sequence[C]//2019 12th International Symposium on Computational Intelligence and Design (ISCID). IEEE, 2019: 285-288.

[17] KARA I. A basic malware analysis method [J]. Computer Fraud & Security, 2019, 2019(6): 11-19.

[18] CWSandbox[EB/OL]. <https://cwsandbox.org/>.



WANG Le-le, born in 1985, Ph.D student. Her main research interests include information security and so on.



WANG Bin-qiang, born in 1963, professor, Ph.D supervisor. His main research interests include network security and broad band information network.