基于语义导向的软件在线升级功能逆向定位

吕小少，舒辉，康绯，黄宇垚

引用本文

吕小少，舒辉，康绯，黄宇垚.基于语义导向的软件在线升级功能逆向定位[J]．计算机科学, 2022, 49(12): 353-361.

LYU Xiao-shao, SHU Hui, KANG Fei, HUANG Yu-yao. Reverse Location of Software Online Upgrade Function Based on Semantic Guidance [J]. Computer Science, 2022, 49(12): 353-361.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.211000059

基于语义导向的软件在线升级功能逆向定位

吕小少　舒　辉　康　绯　黄宇垚
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
(289541163@qq.com)

摘　要　针对软件在线升级的劫持攻击是网络攻击最常用的手段之一。程序分析是快速自动化评估软件升级安全的重要方法,软件中升级功能函数快速逆向定位是实现静态分析和提高动态分析效率的关键前提。传统的程序分析逆向定位,依靠人工经验,根据字符串、API函数等语义信息的交叉引用链关系来实现,效率较低,且无法实现自动化。为解决该问题,提出了一种基于语义分析与逆向分析相结合的软件升级功能定位方法。首先针对软件二进制程序中常见的语义信息(如字符串、函数名、API函数等),建立一个基于自然语言处理的升级语义分类模型;然后借助逆向分析工具提取软件的语义信息,并通过升级语义分类模型来识别升级语义信息;最后定义了一种函数关系调用图形树上的升级函数关键节点求解算法,对升级函数进行求解。文中设计并实现了一个软件在线升级功能定位原型系统,并对常用的153款软件实施了升级功能逆向定位分析,其中126款软件定位成功。通过定位分析初步评估部分软件升级的安全性,获得CNNVD编号漏洞1个,CNVD编号漏洞5个。

关键词:软件在线升级;语义信息;文本分类模型;二进制程序逆向分析;函数定位
中图法分类号　TP393

Reverse Location of Software Online Upgrade Function Based on Semantic Guidance

LYU Xiao-shao,SHU Hui,KANG Fei and HUANG Yu-yao
State Key Laboratory of Mathematical Engineering and Advanced Computing,Information Engineering University,Zhenzhou 450001,China

Abstract　The hijacking attack for software online upgrade is one of the most common methods of network attack. Program analysis is an important method to evaluate the security of software upgrade quickly and automatically. Rapid reverse positioning of upgrade functions in software is a key premise to realize static analysis and improve the efficiency of dynamic analysis. Traditional program analysis reverse localization relies on manual experience based on the cross reference chain relation of semantic information,such as string and API function,which is inefficient and cannot be automated. To solve this problem,this paper proposes a software upgrade function localization method based on semantic analysis and reverse analysis. Firstly,an upgrade semantic classification model based on natural language processing is established for common semantic information(string,function name,API function,etc.) in software binary program. Secondly,the software semantic information is extracted by reverse analysis tool,and the upgrade semantic classification model is used to identify the upgrade semantic information. Finally,an algorithm is defined to solve the key nodes of the upgrade function in the graph tree of function call relationship. This paper designs and implements a software online upgrade positioning system,and carries out reverse positioning analysis on 153 commonly used softwares,126 of which are successfully located. The security of some software upgrades is preliminarily evaluated by positioning analysis,and one CNNVD vulnerability and five CNVD vulnerabilities are found.

Keywords　Software online update,Semantic information,Text classification model,Binary program reverse analysis,Function positioning

1　引言

近年来,利用软件在线升级漏洞的中间人劫持攻击严重威胁着网络空间的安全。2018年12月,黑客团伙利用“驱动人生”系列软件的在线升级通道传播木马下载器,短短两小时就有数万用户遭到攻击。2019年3月,卡巴斯基实验室(Kaspersky Labs)发现,华硕一款Live Update升级软件存在漏洞,被黑客发起了代号为“Operation Shadow Hammer”的攻击,并发起了供应链攻击。一系列利用软件在线升级漏洞的攻击,给用户造成了巨大损失。

程序分析是自动化检测软件升级漏洞的一种有效手段。由于软件功能具有复杂多样性,因此软件升级功能逆向定位

到稿日期:2021-10-10　返修日期:2022-04-23
基金项目:国家重点研发计划“前沿科技创新专项”(2019QY1305)
This work was supported by the National Key R&D Program of China(2019QY1305).
通信作者:舒辉(shuhui123@126.com)

研究是实现该方法的关键前提。针对程序功能逆向定位，通常采取人工逆向分析的方法来完成，效率较低，且无法实现批量软件的自动化分析。由于软件升级漏洞普遍存在，对批量软件的自动化分析成为一个亟待解决的课题。程序静态分析存在无法解决间接寻址的局限性，污点分析和符号执行等动态分析存在效率低下等缺点[1]。因此，基于目前的程序分析方法，还无法很好地实现软件升级功能自动化逆向定位。

随着机器学习和自然语言处理等技术的发展，研究人员越来越多地将人工智能技术应用在逆向分析工作中。大多数逆向分析工作中的语义分析技术，被应用于恶意代码分析和网络协议逆向分析等领域。此前，尚未有研究将语义分析技术和逆向分析技术结合起来，实现对软件功能自动化逆向定位。本文提出的基于语义导向的软件在线升级功能逆向定位，是一种新的逆向分析定位思路。

一个二进制程序可以被看作是一个语义集合的文本，含有字符串、函数名、API 函数等代表各种功能的语义信息。基于这种认识，针对软件升级功能自动化逆向定位问题，本文提出了一种基于语义导向的软件升级功能逆向定位方法，深入剖析软件在线升级的机理和语义信息特点，构建了升级语义信息分类模型，以识别软件中的升级语义信息，并定义了一种函数关系调用树上的关键节点求解算法对升级关键函数进行求解。

本文的主要贡献包括以下 3 个方面：

(1)建立了一种软件在线升级语义分类模型。通过对批量软件在线升级流程的跟踪分析，提取了一批在线升级语义信息样本。在使用自然语言处理技术预处理的基础上将其转换为词向量，并选择分类模型 SVM(Support Vector Machine)对这些词向量进行训练，建立升级语义分类模型。

(2)提出了基于升级语义信息分类预测和关键词匹配的方法，识别软件升级各个阶段的语义信息。通过对软件目录进行遍历，对各可执行文件进行语义信息提取，在预处理后输入软件升级语义分类模型，并将识别出的语义信息根据关键词匹配判断所属的升级阶段，同时，判断出升级模块所在的可执行文件。

(3)定义了一种函数关系调用树上的关键节点求解算法。在已知叶子节点和相关函数调用链的基础上，定义出一种求解共同最近父节点的迭代算法，对函数调用图形树上的关键节点进行求解，计算出关键函数。

2 相关工作

针对软件在线升级安全性问题，研究人员一般是对软件客户端进行动静结合的逆向分析，定位升级模块，理清软件在线升级过程，判断是否存在漏洞。Fu 等[2]采用动静结合的程序逆向分析方法来定位常见的杀毒软件客户端在线升级的代码，并判别是否存在漏洞，发现几款杀毒软件存在在线升级劫持漏洞。火绒团队对 QQ 软件客户端进行逆向分析，定位在线升级模块中的校验代码，发现客户端升级时，仅对升级包的 MD5 值进行校验容易被黑客攻击利用。同时，也有研究者利用软件升级网络流量分析软件的升级流程与安全性，Teng 等[3]提取并分析了软件升级过程中的网络流量，通过模型匹配方法预判升级漏洞，并模拟升级过程进行漏洞验证。上述方法均是人工借助逆向分析工具或者流量分析工具进行手工分析，准确率较高，但是人工分析效率较低，且无法实现对批量软件的自动化分析。对软件升级安全性快速分析的关键前提是自动化逆向定位软件中的升级功能函数。

传统程序分析方法主要分为动态分析和静态分析两种。静态二进制逆向分析存在无法解决的间接寻址问题，研究者们针对该问题做了很多深入的研究，Cifuentes 等[4]利用二进制程序模块划分方法对函数间接调用表进行一种定制化表示，根据一些程序语义特征预测间接调用的目标函数。Kinder 等[5-6]在恢复部分程序控制流图的基础上，对控制流图中的数据关系进行数据流分析，让控制流更加完善，然后在新的控制流图上进行数据流的分析，循环往复，尽量多地恢复程序的控制流信息。这些研究取得了一定的成果，但是无法从根本上解决静态分析存在的天然缺陷。动态分析方法能弥补静态分析的一些不足，但是需要借助动态插桩分析等方法，且对软件在线升级调试环境要求较高，效率较低。由此可见，利用传统的程序分析方法实现对软件升级功能的自动逆向定位面临着较大的困难。

随着人工智能技术的发展，机器学习和自然语言处等技术在程序分析上的应用越来越多。Chua 等[7]利用自然语言处理(Natural Language Processing，NLP)识别二进制程序中的函数特征。Wei 等[8]提出了一种漏洞自动利用的方法，基于 NLP 的信息抽取和基于语义的模糊处理对漏洞利用相关的文本自动生成 POC，发现了 1 个 0day 漏洞和 1 个未披露的漏洞。Bian 等[9]提出了一种利用 NLP 技术推测感兴趣的函数对，成功发现上百对他们感兴趣的函数对，检测出数十个系统错误。自然语言处理在二进制相似性比对上的应用也很普遍，Hu 等[10]提出了一种基于语义的混合方法来比较二值函数相似度，在 Linux 平台上执行 X86，ARM 和 MIPS 体系结构的二进制代码相似性比较。

2018 年，Nan 等[11]介绍了一种 Android APP 中用户敏感信息自动化检测系统——ClueFinder。该方法首先利用关键词匹配自动提取可能出现信息泄露的语义信息(如变量、函数名、参数等)；然后利用自然语言处理技术处理语义信息并训练分类模型，从而自动识别他们感兴趣的元素；最后借助程序分析技术准确识别带敏感信息的应用程序。Nan 等利用该方法对 Google Play 上的 445668 个 APP 做了自动化分析，并取得了较好的效果。虽然该方法针对的是 Android 平台上的 APP 程序，与二进制程序的逆向分析区别较大，但是，该方法的思想对软件在线升级功能逆向定位有一定的借鉴意义。

本文提出的基于软件升级语义信息的自然语言处理方法与程序分析结合的方法是解决软件升级功能逆向定位的一种新思路。

3 软件在线升级功能逆向定位框架设计

本文设计了软件在线升级功能逆向定位方法的整体框架，主要包括 3 个阶段：升级语义信息分类模型训练阶段、升级语义信息自动识别阶段以及函数调用树中升级关键函数求解阶段，框架如图 1 所示。

第一阶段：建立软件升级语义信息分类模型。该阶段包含升级语义信息特征提取及分类模型的训练。跟踪调试一批软件的升级过程，并提取相关的字符串、API 函数、函数名等语义信息作为升级语义学习初始样本；利用自然语言处理技术对初始样本进行预处理并转换为词向量；通过一种半监督的学习方法，选择 SVM 作为分类模型，建立软件升级语义信息分类模型。此外，API 函数作为一种重要语义，通常可以辅助分析程序的功能，本文搜集整理了一个软件升级中常用的 API 函数名库。

第二阶段：软件升级语义信息的自动识别。每个软件大多包含多个可执行程序，升级模块可能在主程序中，也可能是一个单独的程序。本文采取遍历整个软件目录下可执行文件的方法，静态搜索在线升级功能相关语义线索。借助逆向分析工具 IDA Pro[12] 提供的 Python 脚本接口自动提取程序中的字符串、导入函数、函数调用关系等信息；对提取到的字符串函数名等语义信息进行预处理，输入软件升级语义分类模型中进行自动分类，并根据关键字匹配判断是所属的升级阶段。另外，对于提取到的 API 函数，查询第一阶段建立的 API 函数库，并使用 IDA 插件静态获取参数。

第三阶段：升级语义信息函数调用树中升级关键函数节点求解。通过前两个阶段，已获取到了升级模块所在的可执行文件和升级语义信息，借助 IDA Python 编程可以获取升级语义信息的交叉调用函数链。针对升级的各个阶段的关键函数求解问题，本文提出了函数关系调用树上的关键节点求解算法，利用共同最近父节点迭代求解算法对升级各个阶段的关键函数进行求解。

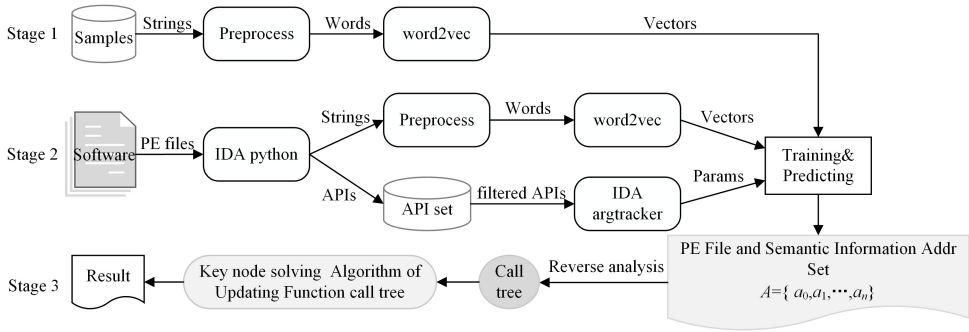

图 1 软件升级功能逆向定位总体架构

Fig. 1 Framework of software updating reverse locating system

3.1 软件升级语义信息特征提取及预处理

在线升级作为软件的一种常见功能，具有特定的流程和语义特征。二进制程序中的语义信息主要包括字符串信息、类名、函数名和 API 函数名等。按软件开发规范，程序中的特征字符串、自定义的类名和函数名代表特定的意义。如图 2 所示，根据某软件的升级功能反编译代码可知，其中存在“Check Update”“app version build=”“Start download update file”等字符串，以及“CheckManualUpdate”等类名，说明了该段代码跟在线升级相关，这些语义信息即为要搜集整理的语义特征样本。

```
v1=(unsigned int)this;
v2=sub_42EF15(&Dst, aUpgrageInfoSEn, (int)L"CheckManualUpdate::
    CheckUpdate");
QQLogOutToFileW2(
&psz,L"QQPlayerUp.exe",
1,
103,
"CheckManualUpdate::CheckUpdate",
v2);
sub_40E404(L"http://upmobile.v.qq.com:1864/Update/app_version_
name=");
LOBYTE(v17)=3;
v6=sub_40E722(L"3.33.3333");
sub_42005F(L"3.33.3333", v6);
v7=sub_40E722(L"&app_version_build=");
```

图 2 某软件在线升级功能反编译代码片段

Fig. 2 Decompiled code of one software upgrade

同时，API 函数是软件另一种重要的语义信息。软件在线升级过程中有网络连接、加解密认证、命令行执行等行为，完成这些行为需要借助特定的 API 函数。例如，软件在线升级常使用 WinInet 或者 Winhttp 库的 API 函数来连接网络，使用 CryptoAPI 函数库中的函数完成加解密，使用 Shellexecute 函数来安装下载的更新包。对于软件在线升级中常用的 API 函数，本文建立了一个常用 API 函数数据库，用于匹配软件中升级相关 API 信息。

对于搜集的升级语义信息样本集，为了训练分类模型，除了 API 函数，本文进一步使用自然语言处理技术处理这些原始语义信息。使用的自然语言处理方法如下：

(1)英文分词

分词操作是将连续字符串拆分成若干个独立的单词或字符。二进制程序中的字符串或者函数名等语义信息常常不是规范的英文写法。有的不是完整的单词，有的单词之间未用空格分开等，如字符串“update_found_new_version”应该被分为“update”“found”“new”和“version”这 4 个英文单词。每一个单词代表一个文本分类维度，用于分类训练。本文借助一种 Python 接口的英文分词工具 wordninja[13]，可以将合在一起的字符串分为一个个小写单词。

(2)去停用词

英文文本中，有些常用词的使用频率相当高，这些词对分析语义信息的意义并无帮助，反而会影响分类的效果，因此要预处理掉。二进制程序中，语义信息中的数字、定冠词等都属于停用词。本文借助自然语言处理工具 NLTK[14] 删除软件语义信息中的停用词。实验证明，去除停用词可以提高升级语义识别精度。

(3)词干提取

词干提取是将英文单词的变体或者衍生体提取为词干或词根等形式的过程,例如,“updating”和“updated”都是升级相关的语义因素,词干提取可以将它们转换为共同的词干“update”。对软件升级相关语义信息而言,词干提取可以将不同词形的单词转换为词根,减少了学习的维度,更加聚焦于软件在线升级功能相关的语义因素,提高文本分类学习的精度。

(4)词嵌入

将样本库中经过预处理后的单词映射到向量空间中,有助于语义相似度的计算。为了自动识别软件程序中在线升级的语义信息,本文利用自然语言处理中的词嵌入技术,将预处理后的语义信息转换为词向量进行分类训练。本文利用最常用的词向量模型,即由 Mikolov 等[15-16]提出的 word2vec 词向量模型,它是一种连续的 skip-gram 模型。word2vec 背后的基本思想是,将在相似上下文中出现的单词映射到相似向量,并且所有的维度都同等重要。此方法适用于软件升级相关语义推断识别。

3.2 升级语义信息分类模型训练

鉴于手工提取的样本只能升级相关正样本,无法简单地进行传统的有监督学习,因此本研究中使用 PU Learning 来解决该问题。

PU Learning[17]是半监督学习[18-19]的一种特殊情况,用于处理在缺少负样本的情况下,利用正样本和无标签样本进行学习训练的问题。PU Learning 分类器的建立一般有直接分类法(Direct Approach)[20]和两步分类法(Two-step Approach)[21]。通过实验比较发现,当正样本数量有限时,两步分类法的分类效果更好。由于手工获取的正样本有限,本研究选择两步分类法。第一阶段,从未标记的样本中筛选出可靠的负样本,具体的实现方式如算法 1 所示;第二阶段,利用正样本和筛选出的负样本,训练传统的监督模型,并进一步预测新的样本。为了提高分类精度,本文选择支持向量机(SVM)模型来标记软件的语义信息是否与在线升级相关。SVM 模型学习一个超平面,将正负向量样本集分割开,以此达到对文本分类的效果。

算法 1 可靠负样本获取算法

输入:正样本数据集 P,未标记数据集 U,间谍样本率 s

输出:可靠负样本集 RN

1. RN←Ø
2. S←Select s% samples from P
   / * 随机从正样本集中取出 s%样本作间谍样本 * /
3. P_S←P-S and set P_S with label 1
4. U_s←U∪S and set U_S with label−1
   / * 剩余正样本集标记为 1,将间谍样本加入未标记样本集,标记为−1 * /
5. g=Generate. Classifier(P_S,U_s)
   / * 利用正负样本集P_S和U_s训练分类器 g * /
6. Pr(d)←g. predict(d),d∈U
   / * 利用分类器 g 对 U 进行分类,得出每一个样本 d 为正的概率 Pr(d) * /
7. Select a threshold θ according to the class-conditional-probability of instances in S
   / * 根据 S 设置负样本概率阈值 * /
8. for d∈U do
9. if Pr(1|d)≤θ,RN=RN∪d
   / * 按阈值取负样本 * /
10. end for
11. Output RN
   / * 输出可靠负样本 * /

在分类训练中,有很多策略可以用来帮助提高 PU Learning。为了进一步提高分类的精度,准确识别在线升级语义信息,本文使用代价敏感分类策略[22],并将损失函数设置为合页损失(Hinge-loss),则正样本分类错误损失目标函数为:

$$F_1=C^{+}_{\sum_{y_i=1}}\max(0,1-y_i(wx_i+b)) \tag{1}$$

将未标记的样本贴上标签,并设定负样本分类错误损失目标函数为:

$$F_2=C^{-}_{\sum_{y_i=-1}}\max(0,1-y_i(wx_i+b)) \tag{2}$$

为使分类更加精准,提高分类错误的代价,设置了 L2 正则化方式,因此本文目标是使以下目标函数最小化:

$$F_1+F_2+\lambda\|w\|^2$$

其中,C^+ 和 C^- 代表对正负分类错误的惩罚因子,在分类训练实践中,C^+ 和 C^- 是根据测试集选择的,因此本文设置 C^+ 远大于 C^-,表明正分类错误代价远大于负分类错误,从而更加准确地识别升级相关语义信息。

通过 PU Learning 训练获取了一批可靠的负样本,加上原本的正样本集,组成了语义信息样本集 S。然后使用 NLP 进行预处理,将 S 转换为一组词向量训练样本集,记为$(X_1,y_1),(X_2,y_2),\cdots,(X_n,y_n)$,其中$\boldsymbol{X}_i$表示特征向量,而 y_i 表示对应的标签。正样本标签为 1,负样本标签为−1。最后将正负样本构建为一组特征向量,将其输入至 SVM 分类器中,训练成一个升级语义分类器,专门用于针对软件在线升级语义信息的自动化识别。在实际工作中,使用 SVM 进行迭代训练[23],对剩余未识别的样本 U 进行分类,分出的负样本再加入到原负样本中进行训练,迭代训练分类器,并通过对测试集分类,观察分类准确率。

3.3 升级语义信息自动识别

经过训练后,分类器可以识别未标记的样本,给定一个软件中的语义信息特征向量,分类器会预测其标签是 1 或者−1,表明该信息分类结果是正相关还是负相关。给定一个软件,遍历其目录下的所有可执行文件,并使用 IDA Pro 脚本提取所有语义信息,除 API 函数外,经过自然语言预处理,形成特征向量,记为 $\boldsymbol{V}(\boldsymbol{S})$。将 $\boldsymbol{V}(\boldsymbol{S})$输入至分类器中,自动识别标签为 1 的语义信息。

API 函数也是软件升级中重要的语义信息,为了更准确地确定某 API 函数是否与升级功能相关,需要获取另一个重要的语义信息,即 API 函数的参数。火眼公司开发的一款名为 argtracker[24]的 IDA 插件,可在静态下对 API 函数的参数进行提取。例如,http 连接 API 函数的参数 WinHttpOpenRequest 的第 3 个参数是网站的 URL,如果有“Update”相关的字串,说明是在线升级的 URL,可以确定为升级相关的

因素。本文方法是将提取的参数输入升级语义分类器中，识别与升级相关的 API 函数。

3.4 基于逆向分析的函数调用关系图形树获取

软件升级大致分为通信阶段和升级包校验安装阶段。本文利用语义信息对升级过程中的关键功能函数进行定位，并依据软件在线升级流程中的语义信息含义将其细分为 4 个阶段：检查新版本、下载升级包、合法性校验和安装软件升级包。流程如图 3 所示。

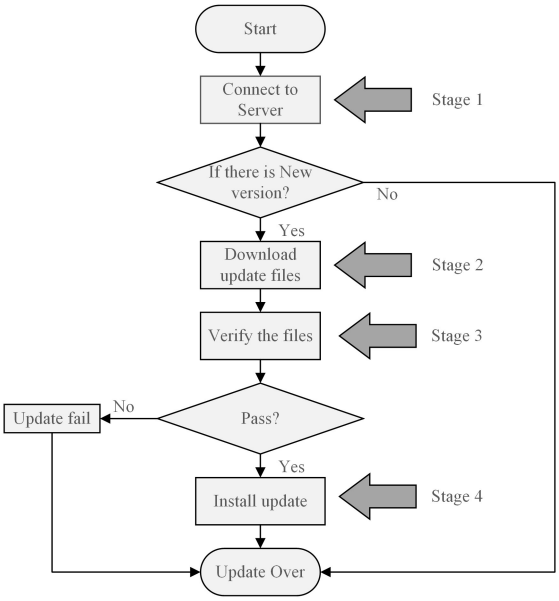

图 3 软件在线升级流程

Fig. 3 Flow of software online upgrade

软件中的语义信息有助于识别其是在线升级中的哪一个阶段。本文使用关键字匹配的方法来确定，其中，API 函数名称也作为一种字符串匹配。各个阶段用于匹配的部分关键字如表 1 所列。

表 1 升级语义信息关键字

Table 1 Keywords for update-related semantics

阶段	关键字
检查新版本	check update, old version, new version, update. ini, update url, etc.
下载升级包	Download, start download, WinhttpRead, InternetReadFile, etc.
合法性检测	Checksum, pubkey, CryptDecrypt, MD5, WinVerifyTrust, etc.
安装升级包	install, update. exe, setup. exe, begininstall, Shellexecute, CreateProcess, WinExec, QProcess: : execute, etc.

通过对前两个阶段的分析，确定升级功能所在程序，并识别出一批该程序里跟升级相关的语义因素，通过 IDA python 获取每个语义因素的交叉引用链。函数调用关系可以用一棵函数关系调用树表示，如图 4 所示的某软件的在线升级调用关系。图中，通过系统前两个阶段对软件的分析，可以识别软件在线升级 4 个阶段的语义信息，并得到它们在程序中的虚拟内存地址。

通过 IDA 脚本对程序进行静态分析，可以获取所有的调用函数链。如图 4 所示，根据关键词匹配，该软件检查版本信息的语义信息“Checkupdate”“app_version_build=”“http://upmobilev.qq.com”这 3 条语义信息的虚拟内存地址均被函数 sub_42F324 调用，可以推断 sub_42F324 函数是检查版本信息的模块。而字符串“StartDownloadTask”跟下载升级包相关，经过分析，被函数 sub_4226D4 调用，而 sub_4226D4 又被函数 sub_42F324 调用，最终推断 sub_42F324 函数的功能是检查版本信息并下载软件更新包。同理，可推断 sub_433F60 函数为升级包合法性检测和执行升级包。需要注意的是，sub_433F60 函数调用链最终并未指向主函数 Winmain，这是因为 sub_432CBC 函数被虚函数调用，虚函数地址在虚函数表中，为间接调用，地址从动态运行中获取，无法被 IDA 静态识别。虚函数是一种常用的面向对象的编程方法，调用地址一般不能直接获取，然而利用本文方法仍然可以定位虚函数表中的校验执行阶段的函数 sub_432CBC。

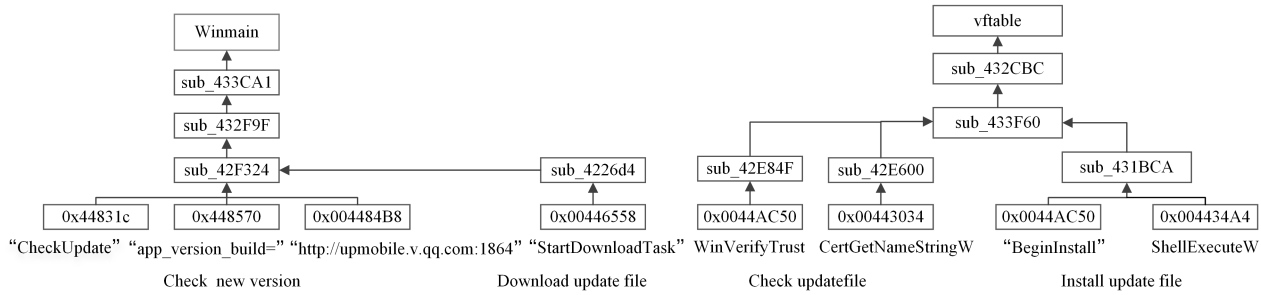

图 4 升级语义信息函数调用图

Fig. 4 Function call diagram of upgrading semantic information

尽管存在间接调用，仍然可以通过语义信息导向的静态分析方法定位到该软件在线升级过程中的下载文件校验和执行阶段。通过程序逆向分析方法可获取每个升级相关语义信息的调用函数链，并且能根据调用链之间的交互关系确定在线升级各个阶段的关键函数。

3.5 函数调用树的关键节点求解算法

本节的最终目的是自动化定位软件在线升级各个阶段的关键函数。对于一个软件，通过升级语义信息分类模型的学习，识别出升级相关的语义信息及其对应的虚拟内存地址，然后利用静态逆向分析技术，提取出这些虚拟内存地址的交叉调用链。这个交叉调用链可以被看作一棵隐形的函数关系图形树，记为 T。本节要解决的关键问题是，从 T 中求解出在线升级各个阶段的关键函数地址。

在软件升级相关函数调用树 T 中，升级相关语义信息为叶子节点。一般来说，升级行为各阶段的语义信息两两间

必然会存在最近的共同父节点，找到相关的共同父节点，即可找到关键函数。但在实际中发现，由于间接调用的存在，有些引用不一定能被静态分析方法找到，且用这种简单的方法找到的共同最近父节点可能不止一个。如何从这些节点中找到最合适的定位函数是需要解决的问题。

本文的解决方法是根据这些语义因素对升级的重要性，对每个因素设置一个权重值 w，并计算父节点调用的叶子节点的个数 n。w 和 n 越高，该节点跟在线升级功能的相关率越高。

定义 1（升级语义信息叶子节点集） 定义软件中升级语义信息交叉调用函数树节点动态集为：$D=\{(a_0,w_0),(a_1,w_1),\cdots,(a_n,w_n),\}$，其中 D 的初始值为所有叶子节点，n 为叶子节点的个数，a_i 为第 i 个叶子节点的内存地址值，w_i 为第 i 个叶子节点升级相关权重值。通过实验，发现权重设置如表 2 所列时的识别效果最好。

表 2 不同语义信息的权重值

Table 2 Weight value of different semantic information

语义类别	权重级别	权重值
能识别参数的 API 函数	高	0.90
字符串（含自定义函数名、类名）	中	0.80
未识别参数的 API 函数	低	0.65

随着不断查找两两节点的共同最近父节点，函数节点集合在不断地进行剪枝。记每个被找到的共同最近父节点向量为 $\boldsymbol{node}=(addr,p)$，其中 $addr$ 是节点地址，p 是该节点的得分值。编号为 i 和 j 的节点的共同最近父节点得分为：

$$node_p=w_i+w_j \tag{4}$$

找到最近共同父节点后，将两个子节点剪掉，用父节点 $node$ 代替。然后递归循环，最终剩下的节点集合即是跟升级相关的节点集，将节点按得分排序，输出得分最高者的地址。

设函数调用树为 T，某升级相关语义信息虚拟内存地址为 a。在 T 上可以求解出 a 的调用函数链集合 S。设 $S=\{C_1,C_2,\cdots,C_m\}$，C_i 为函数链，m 为函数链的个数. 设每条链为 $C=\{f_1,f_2,\cdots,f_n\}$，f_i 为链上的函数，n 为函数的个数。具体求解步骤如算法 2 所示。

算法 2 函数关系调用树升级函数求解算法

输入：函数关系调用树 T，初始升级语义信息节点集 D={(a_0，w_0)，(a_1，w_1)，…，(a_n，w_n)}

输出：升级关键函数地址 a

```
1. S={C1,C2,…,Cm}=T(D),Cx={f1,f2,…,f_count_x}
   /* 获得叶子节点的交叉引用链集合，以及每条函数链上的函数集，count_k为第 x 链节点个数 */
2. for i←1;i≤m;i←i+1 do
3.    for j←i+1;j≤m;j←j+1 do
      /* 取出用于比较的第 i,j 条链 */
4.       for p←1;p≤count_i;p←p+1
5.          for q←1;q≤count_j;q←q+1
         /* 循环取出第 i,j 条链上的函数 */
6.          if C_i.f_p==C_j.f_q do
7.             (a,w)=(C_i.f_p,a_i+a_j)
               /* 找到共同最近父节点，并计算得分 */
8.             del S[i],S[j],D[i],D[j]
                  /* 剪掉两个子节点 */
9.             D.append((a,w))
               /* 将父节点加入节点集 */
10.            S.append({f_P,…,f_count_i})
               /* 父节点链加入链集合中，做递归循环 */
11.      end if
12.         end for
13.         end for
14.      end for
15. end for
16. D.w_n=max(D.w_0,D.w_1,…D.w_m)
    /* 剩余节点比较得分大小，取得分最高者 */
17. return D.a_n
```

4 实验设计与实现

本节主要介绍软件升级定位原型系统的实现过程，并利用该系统对实际软件升级功能进行逆向定位分析。

4.1 实验设置

本文使用 Python 语言和 IDA Pro 实现本文的原型系统。原型系统分为两个子系统：软件升级语义识别子系统和软件升级功能关键函数求解子系统。其中，语义分类识别子系统使用机器学习库 Scikit-learn 机器学习框架运行 SVM 分类的 Python 代码来训练升级语义分类器。软件升级功能关键函数求解子系统使用 IDA Python 脚本实现，软件版本为 IDA7.5 和 Python 3.7.5，实验在 VMware 下安装的 Win7 系统下进行。

实验对象是通过腾讯软件管家、360 软件管家等软件安装工具安装的一批常用的软件。共在 Win7 虚拟机下安装了 8 类共 183 款软件，如图 5 所示。

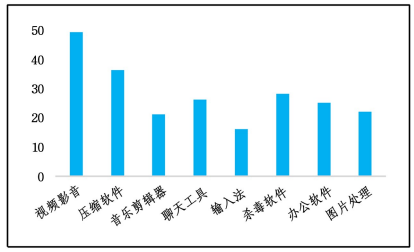

图 5 实验样例分布图

Fig. 5 Distribution diagram of experimental samples

4.2 实验结果

本实验主要包括软件升级语义分类训练识别和软件升级关键函数定位两部分。

（1）软件升级语义分类器训练

分类器样本从已安装的软件中随机挑选出的 30 款软件中获取。通过人工逆向分析，获取正向样本集共 436 条，包括字符串、类名、函数名等；自动提取语义信息 2 353 条，作为未标记数据集。同时，提取了 79 个 API 函数，组成升级 API 函数数据库，用于匹配。

在两阶段策略 PU Learning 训练分类模型中，第一阶段从正样本中选取 10%的样本作为间谍样本，加入未标记的样本中，标记为负样本，并使用可靠负样本获取算法获取可靠

负样本，概率阈值设置为 15%。第二阶段首先将正样本集 P 和获取的负样本 RN 按 8∶2 的比例分为训练集和测试集。将训练集使用 SVM 模型训练成一个分类器，并用分类器对未识别的样本进行分类，将识别出的负样本加入 RN 中，然后继续迭代训练。随着负样本的增多，测试集的识别精度变化如图 6 所示。

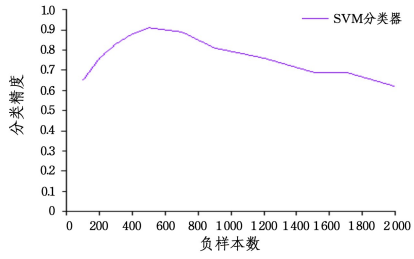

图 6 不同数量负样本的训练精度

Fig. 6 Training accuracy with different number of negative samples

可以看出，当负样本较少时，分类精度较低；随着负样本增多，精度变高。当负样本数量跟正样本数量相近时，分类器的分类效果最好；当负样本数量明显多于正样本时，精度开始下降。分类效果最好时，准确率为 91%，说明此方法能以较高概率识别软件升级语义信息。

(2)软件升级关键函数定位

利用软件升级语义分类器对剩余的 153 款软件的升级语义信息进行识别，找出升级功能所在的程序以及语义信息，借助逆向分析手段和函数调用树上的关键节点求解算法，进行在线升级功能的关键模块定位分析，成功定位了 127 款软件，有 26 款软件定位失败。

定位失败的软件中，19 款软件程序存在代码混淆等方法保护，静态方法无法对其进行分析，不属于本系统分析的范围内。另有 7 款软件中升级语义信息不足，系统未能识别到足够的升级语义信息，因此未能成功进行定位分析。为了验证本文方法的准确性和可靠性，对其中 10 款软件进行了在线升级动态调试，具体结果如表 3 所列(虚拟内存基地址均为 0x401000)。

表 3 部分实验分析结果

Table 3 Part of experimental analysis results

软件	升级程序	关键函数	定位是否精准
QQ 影音 4.6.3	QQPlayerUP.exe	sub_42F324(检查更新和下载更新包) sub_433F60(合法性检测和安装更新包)	是
美图秀秀 4.0.1	LiveUpdate.exe	sub_478690(检查更新和下载更新包) sub_478690(合法性检测) loc_484ADB(安装更新包)	是
NovelReader 2.15	NRUpdater.exe	sub_408A80(检查更新) sub_40A580(下载更新包) sub_408650(安装更新包)	是
游侠日历 2.1.0	update.exe	sub_4B6400(检查更新) sub_4B8050(下载更新包) sub_4B7C90(安装更新包)	是
铠甲卫士 1.0.16	Kjupdate.exe	sub_40FDC0(检查更新) sub_4112E0(下载更新包) sub_411C90(合法性检测) sub_415DC0(安装更新包)	是
Poedit 2.2.4	Poedit.exe	sub_5B2C20(检查更新) sub_4A9F90(下载更新包)	是
人人影视 3.1.0	RRShare.exe	sub_431CC0(检查更新和下载更新包) sub_495d00(安装更新包)	是
桌面日志 9.5.2	DesktopNotes.exe	sub_551680(检查更新) sub_457A50(下载更新包) sub_457A50(安装更新包)	是
SpritePDF 1.0.0.0	Update.exe	sub_404720(下载更新包) sub_4048F0(安装更新包)	是
CloudMail 1.0.01	upgrade.exe	sub_4071B0(检查更新) sub_4071B0(下载更新包) sub_405990(安装更新包)	是

从实验中可以看到，针对其中几款软件，系统未识别出升级包合法性检测的升级流程阶段，且通信使用 HTTP 的明文传输协议，说明其不存在升级包校验功能，直接安装更新包，可以初步判断为存在安全漏洞。因此，通过对软件升级功能逆向定位，可以初步评估软件在线升级是否安全。

4.3 对比分析

传统的逆向分析中，对程序中关键代码的定位一般依赖于人工经验和程序代码中的 API 函数、字符串等语义信息进行综合判断，同时也会结合动态分析设置断点调试的方法。这种方法的准确率较高，但是效率较低，不适用于自动化程序分析。

据统计，本文的基于语义分析的方法绝大部分都能在 20 s 内完成分析，大部分时间是用于 IDA Pro 对软件中可执行文件的反编译解析，相比传统手工分析，效率大大提高。人工分析在借助动态调试的基础上，能完全保证定位的正确性。原型系统属于自动化静态分析，由于软件编写的多样性和复杂性，准确率不可能达到 100%。本文对分析成功的 50 款软件进行调试验证，发现有 46 款软件分析准确，准确率达到

92%，基本满足分析需要。

4.4 案例分析

本小节针对一款小说阅读软件 NovelReader 2.15 做具体案例分析。为了验证本文方法的准确性和可靠性，原型系统详细记录了分析日志；同时，利用手工逆向方法，对该软件的在线升级过程进行跟踪，通过比对结果来判断系统自动分析的准确性。

原型系统首先遍历软件目录，调用 IDA python 脚本自动提取软件里的字符串、函数名、API 函数等语义信息，经过升级分类模型自动识别出升级语义信息，并确定了跟升级相关的程序是名为 NRUpdater.exe 的文件。各个阶段的部分关键语义信息如表 4 所列。

表 4 NovelReader 2.15 升级语义信息

Table 4 NovelReader 2.15 update semantic information

升级阶段	语义信息
检查更新	"http://api.xiaoq8.com/tbook.php/Api/Api/update?ver=%d" http://api.xiaoq8.com/tbook.php/Api/updatedatabase?vver= "MCCheckUpdate::'vftable'"
下载	"AUIDownloadManager" WinHttpReadData
校验	无
安装	ShellexecuteW,"setup.exe"

版本检查阶段匹配出相关的类名语义信息"MCCheckUpdate"和两个升级 URL 地址。系统定位的版本查询函数为 sub_408A80。如图 7 所示，sub_408A80 首先获取 MCCheckUpdate 类的虚函数表，sub_4193F0 函数访问 URL 检查软件更新版本号。为确认该定位的准确性，直接在该函数处设置断点后动态调试观察，程序会在 sub_4193F0 函数中访问软件升级 URL 并检查是否有新版本。采用相同的方法，针对软件升级包下载阶段和升级包安装执行阶段，进行了动静结合的手动分析。通过与本文的原型系统的分析日志进行对比，证实了原型系统定位的关键函数是准确的。

```
int __userpurge sub_4046C0@<eax>(int result@<eax>, int a2)
{
  *(__DWORD *)result=&GameMgr::MCCheckUpdate::'vftable';
  *(__DWORD *)(result+20)=0;
  v14=1;
  sub_4193F0(&v11, L("http://api.xiao8.com/tbook.php/update?ver
  =%d",11);
  v1=operator new(0x294u);
}
```

图 7 版本查询阶段反编译代码片段

Fig. 7 Decompiled code of check updating

本文的原型系统未检测出该软件在线升级过程中有软件包校验阶段，并且通信协议为 HTTP 明文通信，初步可以判断该软件的在线升级存在安全风险。在模拟环境下，对该软件的在线升级过程进行劫持，成功实施了中间人攻击。

通过定位软件升级功能，原型系统梳理了升级流程，并在静态下初步评估软件在线升级的安全性。通过系统自动评估和模拟攻击确认，发现了软件升级劫持漏洞若干，并申请 CNNVD 漏洞编号 1 个，CNVD 漏洞编号 5 个，结果如表 5 所列。

表 5 软件升级漏洞编号

Table 5 Software upgrade vulnerability number

软件	版本	漏洞编号
央视影音	4.4.0.0	CNNVD-201903-1283
CAJViewer	7.2.113.0	CNVD-2018-02906
美图秀秀	3.1.6.1003	CNVD-2018-06126
迅雷	9.1.21.536	CNVD-2018-06136
酷我音乐盒	8.7.4.0	CNVD-2018-06296
优酷	7.2.6.8110	CNVD-2018-06299

结束语 本文提出了一种基于语义导向的软件升级功能自动逆向定位的新思路，借助自然语言处理和机器学习方法，对软件在线升级语义信息建立了分类模型，以期自动预测软件中与升级相关的语义信息。在此基础上，定义了一种函数调用关系树上的升级关键函数节点求解法，求解出与升级相关的函数。本研究完成了原型系统，通过对 153 个软件的升级功能进行分析，对其中的 127 款完成升级功能逆向定位，证明了所提方法和系统能够准确地对软件升级功能进行逆向定位。

参 考 文 献

[1] ZHANG J, ZHANG C, XUAN J F, et al. Research Progress of Program Analysis[J]. Journal of Software, 2019, 30(1): 80-109.

[2] FU J M, LIU G, LI P W, et al. A security analysis method for antivirus software upgrade process [J]. Journal of Wuhan University(Science Edition), 2015(12): 509-516.

[3] TENG J H, GUANG Y, SHU H, et al. Automatic Detection Method of Software Upgrade Vulnerability Based on Traffic Analysis[J]. Journal of Network and Information Security, 2020, 6(1): 94-108.

[4] CIFUENTES C, MIKE V. Recovery of jump table case statements from binary code[J]. Science of Computer Programming, 1999, 40(10): 171-188.

[5] KINDER J. Static Analysis of x86 Executables[D]. Darmstadt: Technische Universitat Darmstadt, 2010.

[6] KINDER J, VEITH H. JAKSTAB. A Static Analysis Platform for Binaries[C]// Proceedings of the 20th International Conference on Computer Aided Verification. 2008: 423-427.

[7] CHUA Z, SHEN S, SAXENA P, et al. Neural nets can learn function type signatures from binaries[C]// Proceedings of the USENIX Security. 2017: 99-116.

[8] WEI Y, ZONG P, CHEN K, et al. SemFuzz: Semantics-based Automatic Generation of Proof-of-Concept Exploits[C]// ACM SIGSAC Conference on Computer and Communications Security. 2017: 2139-2154.

[9] BIAN P, LIANG B, HUANG J, et al. SinkFinder: Harvesting Hundreds of Unknown Interesting Function Pairs with Just One Seed[C]// Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 2020: 1101-1113.

[10] HU Y, WANG H, ZHANG Y, et al. A Semantics-Based Hybrid Approach on Binary Code Similarity Comparison [J]. IEEE Transactions on Software Engineering, 2021(6): 1241-1258.

[11] NAN Y H, YANG Z, WANG X, et al. Finding Clues for Your Secrets: Semantics-Driven, Learning-Based Privacy Discovery in Mobile Apps[C]// Proceedings of the 24th Annual Network and

Distributed System Security Symposium. 2018.

[12] DUAN G. Encryption and decryption [M]. Publishing House of Electronics Industry. 2018：65-94.

[13] DEREK A. Wordninja [EB/OL]. https：//github. com/keredson/wordninja.

[14] NLTK. NLTK Document[EB/OL]. http：//www. nltk. org/.

[15] MIKOLOV T，CHEN K，CORRADO G，et al. Efficient estimation of word representations in vector space[C]// Proceedings of the International Conference on Learning Representations(ICLR 2013). 2013：1-12.

[16] MIKOLOV T，SUTSKEVER I，CHEN K，et al. Distributed Representations of Words and Phrases and Their Compositionality[C]// Proceedings of the Advances in Neural Information Processing Systems. 2013：3111-3119.

[17] CHARLES E，KEITH N. Learning Classifiers from Only Positive and Unlabeled Data[C]// Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2008：213-220.

[18] OLIVIER C，BERNHARD S，ALEXANDER Z. Semi-Supervised Learning[J]. IEEE Transactions on Neural Networks，2009，20(3)：542-542.

[19] ZHOU Z H，LI M. Semi-Supervised Learning by Disagreement [J]. Knowledge and Information Systems，2010，24(3)：415-439.

[20] CHARLES E，KEITH N. Learning Classifiers from Only Positive and Unlabeled Data[C]// Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2008：213-220.

[21] KABOUTARI A，BAGHERZADEH J. An Evaluation of Two-Step Techniques for Positive-Unlabeled Learning in Text Classification[J]. International Journal of Computer Applications Technology and Research，2014，3(9)：592-594.

[22] MARTHINUS C，GANG N，MASASHI S. Analysis of Learning from Positive and Unlabeled Data[C]// Advances in Neural Information Processing Systems. 2014：703-711.

[23] HWANJO Y，JIAWEI H，CHANG K. PEBL：Web page classification without negative examples[J]. IEEE Transactions on Knowledge and Data Engineering，2004，16(1)：70-81.

[24] FLARE. IDA Pro Script Series：Automating Function Argument Extraction[EB/OL]. https：//www. fireeye. com/blog/threat-research/2015/11/flare_ida_pro_script. html.

LYU Xiao-shao，born in 1989，postgraduate. His main research interests include cyber security and reverse engineering.

SHU Hui，born in 1974，Ph. D，professor，Ph.D supervisor. His main research interests include cyber security and reverse engineering.

（责任编辑：何杨）