

Excel 4.0 宏自动化反混淆与家族分类系统

李晨光, 杨秀璋, 彭国军

(武汉大学国家网络安全学院, 湖北 武汉 430072)

摘要: 近年来, 使用恶意 Excel 4.0 宏 (XLM) 文档的攻击迎来了爆发, 而 XLM 代码往往经过复杂的混淆, 现有方法或检测系统难以分析海量样本的真实功能。因此, 针对恶意样本中使用的各类混淆技术, 基于抽象语法树和模拟执行, 设计和实现了包含 138 个宏函数处理程序的自动化 XLM 反混淆与关键威胁指标 (IOC, indicators of compromise) 提取系统 XLMRevealer; 在此基础上, 根据 XLM 代码特点提取 Word 和 Token 特征, 通过特征融合能够捕获多层次细粒度特征, 并在 XLMRevealer 中构造 CNN-BiLSTM (convolution neural network-bidirectional long short term memory) 模型, 从不同维度挖掘家族样本的关联性和完成家族分类。最后, 从 5 个来源构建包含 2 346 个样本的数据集并用于反混淆实验和家族分类实验。实验结果表明, XLMRevealer 的反混淆成功率达到 71.3%, 相比 XLMMacroDeobfuscator 和 SYMBEXCEL 工具分别提高了 20.8% 和 15.8%; 反混淆效率稳定, 平均耗时仅为 0.512 s。XLMRevealer 对去混淆 XLM 代码的家族分类准确率高达 94.88%, 效果优于所有基线模型, 有效体现 Word 和 Token 特征融合的优势。此外, 为探索反混淆对家族分类的影响, 并考虑不同家族使用的混淆技术可能有所不同, 模型会识别到混淆技术的特征, 分别对反混淆前和反混淆后再统一混淆的 XLM 代码进行实验, 家族分类准确率为 89.58%、53.61%, 证明模型能够学习混淆技术特征, 更验证了反混淆对家族分类极大的促进作用。

关键词: 恶意宏文档; Excel 4.0 宏; 反混淆; 家族分类

中图分类号: TP393

文献标志码: A

DOI: 10.11959/issn.2096-109x.2024040

Automated deobfuscation and family classification system for Excel 4.0 macros

LI Chenguang, YANG Xiuzhang, PENG Guojun

School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China

Abstract: In recent years, a surge has been witnessed in cyber-attacks that leverage malicious Excel 4.0 macros

收稿日期: 2023-12-26; 修回日期: 2024-04-10

通信作者: 彭国军, guojpeng@whu.edu.cn

基金项目: 国家自然科学基金 (62172308, U1636107, 61972297, 62172144), 中央网信办网络安全学院学生创新资助计划

Foundation Items: The National Natural Science Foundation of China (62172308, U1626107, 61972297, 62172144), Cyber Security School Student Innovation Funding Program of Office of the Central Cyberspace Affairs Commission

引用格式: 李晨光, 杨秀璋, 彭国军. Excel 4.0 宏自动化反混淆与家族分类系统[J]. 网络与信息安全学报, 2024, 10(3): 66-80.

Citation Format: LI C G, YANG X Z, PENG G J. Automated deobfuscation and family classification system for Excel 4.0 macros[J]. Chinese Journal of Network and Information Security, 2024, 10(3): 66-80.

(XLM) within documents. Malicious XLM codes often undergo complex obfuscation, posing a substantial challenge for conventional analysis methods and detection systems to discern the actual functionality within a vast array of samples. Consequently, an automated system for deobfuscating XLM and extracting key Indicators of Compromise (IOCs), named XLMRevealer, was developed to counter the diverse obfuscation strategies employed in malicious samples. XLMRevealer was architected upon abstract syntax trees and execution simulation, encompassing 138 comprehensive macro function handlers. Based on that, Word and Token features tailored to XLM code peculiarities were extracted, capturing multi-level, fine-grained features through feature fusion. XLMRevealer incorporated a CNNBiLSTM model to discern familial correlations across dimensions, facilitating family classification. Finally, a dataset comprising 2346 samples from five distinct sources was constructed for both deobfuscation and family classification experiments. Results indicated that XLMRevealer achieved a 71.3% deobfuscation success rate, outperforming XLMMacroDeobfuscator and SYMBEXCEL by 20.8% and 15.8%, respectively. Its efficiency was stable, with an average processing time of only 0.512 seconds. The family classification accuracy for deobfuscated XLM codes stood at 94.88%, surpassing all baseline models and underscoring the efficacy of Word and Token feature integration. Furthermore, to assess the impact of deobfuscation on family classification and account for variability in obfuscation techniques across families, experiments were conducted on both the original and uniformly obfuscated XLM codes. The accuracies were 89.58% and 53.61%, respectively, demonstrating the model's capability to learn obfuscation features and confirming the significant enhancement deobfuscation provides for family classification.

Keywords: malicious macro document, Excel 4.0 macro, deobfuscation, family classification

0 引言

在网络攻击的杀伤链模型中,将恶意代码投递到受害者机器中是最重要的步骤之一,而伪装恶意代码及其载体从而欺骗受害者下载、运行攻击负载是攻击成功的关键。针对上述问题,滥用 Microsoft (MS) Office 文档作为恶意代码载体的攻击方式逐渐出现,并被各攻击组织利用^[1]。

MS Office 具有自动运行并完成复杂任务的宏功能,不仅能实现文档操作,还能与操作系统、网络进行交互。攻击者常通过社会工程学手段,利用新闻热点或针对特定目标画像伪装恶意宏文档,以诱使目标运行宏。基于此,利用钓鱼邮件投递恶意宏文档并欺骗受害者运行的攻击仍是网络钓鱼攻击的重要组成部分^[2]。

常被用于执行恶意行为的宏中最流行的应是 VBA (visual basic for application) 宏,自 1995 年第一个宏病毒“Concept”被发现并大范围传播,以 MS Office 文档为载体的恶意 VBA 宏代码开始广泛流行。目前,研究人员已经在 VBA 宏代码的恶意性检测与反混淆方面进行了深入研究并取得系列成果^[3-6]。为了更好地实施高隐蔽、高对抗的宏代码攻击,攻击者将目光转向 Excel 4.0 宏。

XLM 由 Microsoft 于 1992 年发布并应用于 Excel 4.0,作为 VBA 的前身,其功能强大,且

至今仍受 MS Office 支持,而先前鲜有攻击者利用 XLM 制作恶意宏文档,导致各类反病毒软件缺乏对恶意 XLM 代码的检测,种种因素使恶意 XLM 文档的流行成为可能。随着网络安全厂商 Outflank 的安全研究员首次公开使用 XLM 执行 ShellCode 的利用代码,利用恶意 XLM 文档的网络钓鱼攻击在 2020 年迎来了爆发,在很长一段时间内,恶意 XLM 文档几乎能够躲避所有反病毒软件的扫描。

虽然 XLM 与普通 Excel 公式(或函数)的使用方法类似,语法十分简单,但其特性还是给 XLM 分析带来了挑战。

1) 攻击者尝试各种规避手段来处理 Excel 文档,以阻碍安全分析人员直接查看文件中的恶意宏代码,如深隐藏宏表(只能通过修改对应二进制字段值取消隐藏)、在较大行列数处存储代码、将字体颜色改为背景色、保护宏表单元格等,这要求解析文档能直接准确地提取出宏代码。

2) 有多种 MS Excel 支持的复杂文件格式能够插入 XLM 表并运行宏,分析工具需要解析各种文件格式从而正确提取 XLM。

3) 不断演变的 XLM 代码混淆方法能够有效阻碍安全人员分析恶意代码并提取其中的 IOC (indicators of compromise),真实样本中的 XLM

代码往往经过多层混淆,且包含的公式数量庞大,人工分析出混淆代码的真实功能意图几乎是无法完成的任务。

正确提取并反混淆 XLM 代码对分析相关攻击具有至关重要的作用。一方面,想要了解样本所执行的具体恶意行为,则必须去除混淆的繁杂无意义代码,提取恶意功能代码。另一方面,为了便于后续对类似样本的检测、分析,防止未来可能的攻击感染,必须从反混淆后的代码和数据中提取威胁指标^[1]。

此外,在真实世界对黑客攻击的检测与分析中,面对海量样本时无法逐一分析,这就要求划分样本类别,再对各类样本进行抽样分析或批量提取 IOC。总之,确定恶意样本所属家族类别有助于安全人员快速分析恶意样本的主要功能、定位并提取关键特征,从而更好地实现黑客攻击溯源和攻击者画像分析。因此,针对 XLM 的家族分类研究具有十分重要的意义。而重混淆代码包含大量仅用于混淆的无意义数据和代码,势必影响家族分类效果,反混淆是否对家族分类有促进作用也有待研究。

针对上述背景,本文将开展 XLM 反混淆与家族分类研究,本文主要贡献如下。

1) 设计并实现 XLMRevealer,能够准确提取并反混淆 XLM 代码。提出一种融合抽象语法树 (AST, abstract syntax tree)、模拟执行、正则匹配等的反混淆集成方法,实现 138 个宏函数的模拟执行程序 (优于现有最好工具的 90 个函数)。本文系统地分析恶意 XLM 代码混淆手段并完善模拟执行流程,所实现方法能有效对抗各种抗文件解析手段,并具有较强的鲁棒性和反混淆能力。

2) 针对 XLM 代码特点,设计并实现了两种特征 (Word 级与 Token 级) 的提取、嵌入和特征融合方法,能够获取 XLM 多层次细粒度特征,并构建基于 CNN-BiLSTM (convolution neural network-bidirectional long short term memory) 的 XLM 家族分类模型。该模型能够有效关联各种特征嵌入、捕获多层次语义关系,以精确识别恶意家族。

3) 结合安全公司公开数据集、网络恶意样

本数据库等 5 个数据来源构建了包含 2 346 个恶意 XLM 样本的数据集,并开展详细的反混淆实验与反混淆前后的家族分类实验。

4) 阐述了反混淆的基本原理,对前期调研和实验评估中观察到的真实世界攻击样本所采用的混淆方法进行全面概述,深入讨论混淆原理及对应的反混淆方法。

1 相关工作及研究动机

本节主要概述 Excel 4.0 宏安全威胁的基本背景,以及针对 XLM 提取与反混淆的研究现状。

1.1 Excel 4.0 宏及安全威胁

1.1.1 Excel 4.0 宏

XLM 诞生至今已有三十多年,而 Microsoft 仍支持该宏,且能存储并运行 XLM 的 4 种文件格式 (xls、xlsb、xlsm、slk),这为攻击者伪装恶意 XLM 文档进行钓鱼攻击提供了便利。

XLM 代码必须存储在 Excel 4.0 宏表单元格中,一个单元格存储一条 XLM 代码或数据,笔者将一条 XLM 代码称为公式 (formula)。公式可细分为宏函数调用公式、变量操作公式 (数学计算、字符串操作等)、自定义函数调用公式、赋值公式等,除了赋值公式外,所有公式均必须以等号 “=” 起始。

一般情况下,XLM 从一个具有 “Auto” 前缀定义名称 (如 “Auto_Open”) 的单元格开始,在一列中按照从上到下的顺序运行,直到遇到终止公式 “=HALT()”,也能由 “=GOTO(ref)” “=R1C2()” 等公式改变控制流实现跳转。

宏函数数量繁多且功能强大,由等号和宏函数构成的宏函数调用公式能用于实现主要恶意功能和混淆,如 “=EXEC(“calc.exe”)”。此外,赋值公式中的函数自定义公式由自定义函数名、等号、自定义函数起始公式所在单元格地址构成,如 “runcalc=R1C2”,后续通过自定义函数调用公式 (如 “=R1C2()”、“=runcalc()”) 调用创建的函数,自定义函数段以 “=RETURN()” 结尾。

1.1.2 XLM 混淆技术

XLM 混淆技术主要有以下几种,具体实例如图 1 所示。

(1) CHAR 与数值运算函数

“CHAR(num)” 返回十进制数 num 对应的

ASCII字符, 通过将重要字符串(如URL、shell命令)或XLM公式的各字符转换成十进制数作为参数, 能有效隐藏真实数据和代码。其参数也能由“XOR”、“SUM”等数值运算函数通过计算一些无意义数值得到, 从而加深混淆程度。

(2) 字符串操作

“CONCATENATE”与“&”操作符都可用来拼接字符串, “MID”能够截取给定字符串中的子串等。字符串操作函数与“INT(字符转整数)”“CHAR”配合使用实现多层混淆。

(3) 日期信息

“DAY(NOW())”获取当前时间在该月的第几天, 将该值输入数值运算函数中并配合“CHAR”等函数使用, 使XLM代码只有在每个月中攻击者指定的那天才能正常运行。

(4) 执行环境信息

系列“GET”类函数能够获取Excel以及操作系统的各种信息。如“GET.CELL”获取单元格的属性值、“GET.DOCUMENT”获取工作簿中表信息等, 这些信息可用于混淆和环境感知。

(5) 自定义函数名

赋值公式能够实现函数自定义, 如图1中R1C1单元格公式所示。此外, “REGISTER”能注册DLL(动态链接库, dynamic link library)中的导出函数并自定义一个别名用于调用。大量自定义函数名的使用能隐藏代码的真实功能。

(6) 数据流混淆

使用“SET.NAME”或赋值公式设置随机变量名并赋值很大程度扰乱分析人员对关键数据的追踪; 使用“FORMULA”向单元格中填充数据或公式, 再通过单元格的引用调用其内容; 此外, 很多样本会调用大量的“FWRITE”将后续攻击负载内容逐字符写入文件。滥用数据流混淆能够深度隐藏真实代码与重要IOC。

(7) 控制流混淆

调用“GOTO”、“RUN”以及自定义函数能将控制流转到目标单元格; “FOR”“FOR.CELL”“WHILE”分别与“NEXT”搭配使用实现循环, 常用于构造后续攻击负载代码。滥用控制流混淆能使安全人员无法确认执行真正恶意功能的代码段, 让代码功能分析及关键数据提取变得困难。

(8) 一个单元格内多条公式

虽然MS在Excel公式用法的官方文档中明确说明一个单元格只能写入一条公式, XLM实际却支持运行由等号连接的多条公式。该混淆方法可能会导致反混淆时抽象语法树解析错误, 并使模拟执行略过部分公式。

```
①CHAR与数值运算函数
=CHAR(BITXOR(72,117)) | =CHAR(SUM(27,42))
=CHAR(MOD(123,62))
②字符串操作
=CONCATENATE("&"&"EXEC(", CHAR(34),
MID("ascalc.exeze",3,8), CHAR(INT("4"&"1"))))
③日期信息
=CHAR(DAY(NOW)+31)
④执行环境信息
=IF(GET.WINDOW(7),GOTO(R1C2),)
=CHAR(GET.WORKSPACE(42)+60)
⑤自定义函数名
[R1C1] sLJeWzM=R5C5 [R5C5] =EXEC("calc.exe")
[R2C1] =sLJeWzM() [R6C5] =RETURN()
=REGISTER("User","GetTickCount","J","GTC")
⑥数据流混淆
[R3C4]=SET.NAME("cmd","calc.exe") / cmd="calc.exe"
[R4C4]=EXEC(cmd)
=FORMULA("=",R1C2)
⑦控制流混淆
[R1C1] =GOTO(R6C1) [R2C3] =WHILE(num<5)
[R6C1] =RUN(R30,C40) [R6C3] =NEXT
⑧一个单元格内多条公式
[R1C1] =ALERT(1)=ALERT(2)=EXEC("calc.exe")
```

图1 混淆技术实例

Figure 1 Examples of obfuscation techniques

1.1.3 安全威胁

网络钓鱼是主要的网络攻击之一, 是社会工程学最常见的一种形式^[7], 而利用伪装的恶意宏文档作为钓鱼邮件附件发起的攻击更是屡见不鲜, 这种攻击不仅能利用社会热点进行伪装, 从而发起大范围的网络钓鱼, 还能通过分析目标受害者画像精心构造邮件及附件文档内容, 从而提高攻击成功率。且XLM能够满足攻击者的各种需求。因此, 利用恶意XLM文档的钓鱼邮件攻击给全球网络安全带来极大的安全隐患。

XLM虽然语法简单, 却有复杂多样的混淆手段, 即便被检测到并提取出代码, 仍能够抵抗安全人员分析其真实功能和意图。此外, 在网络空间安全对抗中, 数量巨大的攻击样本和不断涌

现的新攻击手段让安全人员迫切需要恶意样本家族分类技术,以减轻样本分析负担,并利用恶意样本间的同源关系进行网络攻击溯源,从而进一步追究和震慑网络攻击组织^[8]。因此,实现有效的 XLM 提取与反混淆工具,并构建模型来识别 XLM 恶意家族至关重要。

1.2 研究现状

针对恶意 XLM 代码检测、提取与反混淆的现有研究与本文工作对比如表 1 所示。

XLMMMD 和 SYMBEXCEL 都无法支持 .slk 文件,其模拟执行模块所支持的函数数量较少、没有完全覆盖攻击样本中常用的各种函数,对定义名称引用、单元格地址引用、变量自定义、单元格内多公式等混淆手段的反混淆效果较差,反混淆结果保留了大量用于混淆的中间信息,且无法提取重要 IOC。

此外,现有研究仅是提取反混淆 XLM 代码,缺少对恶意 XLM 家族的分类研究,而结合了机器学习与深度学习的恶意性 VBA 检测,恶意 PowerShell 判定方法具有一定参考性。

在恶意性 VBA 检测方面,Miura 等^[3]利用词频(TF, term frequency)和词频-逆文件频率(TF-IDF, term frequency-inverse document frequency)去除不重要的词,通过 Doc2Vec^[9]提取特征向量,最后使用支持向量机等机器学习模型判定恶意性;Mimura 等^[6]基于自然语言处理的方法,将 VBA 分解为单词,使用 TF 或 TF-IDF 选择重要词,利用词袋实现 Doc2Vec 以提取特征向量,最后使用随机森林等判断恶意性。

在恶意 PowerShell 检测方面,2018 年,Hen-

dlar 等^[10]利用 n -gram 分词提取代码特征,使用卷积神经网络进行恶意性判定;2020 年,Hendler 等^[11]又提出了基于上下文词嵌入的深度学习检测方法,利用 Word2Vec^[12]等方法提取上下文词嵌入信息,输入卷积神经网络等实现判定;Tajiri 等^[13]使用词袋、Doc2Vec 和潜在予以索引提取特征,利用支持向量机、XGBoost 等作为分类器判定恶意性。

2 系统设计

XLMMRevealer 由 XLM 代码提取模块、反混淆模块、家族识别模块 3 部分构成,XLMMRevealer 系统框架图 2 所示。

2.1 XLM 代码提取模块

XLM 代码提取模块负责解析文件并提取所有宏表及工作表、存储了公式或数据的单元格、定义名称(及对应单元格地址)、XLM 运行入口点。该模块提取定义名称为“Auto_Activate”“Auto_Deactivate”“Auto_Open”“Auto_Close”的单元格作为运行入口点。为了应对利用文档信息的混淆技术,还会提取出单元格属性、注释等信息。该模块在充分解析 Excel 文档后,利用提取的重要数据创建单元格类、表类实例,供后续反混淆使用。

XLMMRevealer 支持两种文件解析方式:静态文件解析、MS COM(组件对象模型,component object model)加载文件解析。

1) 静态文件解析:XLMMRevealer 读取文件二进制数据,基于 BIFF、XML、SYLK 文件结构及字段意义进行定义,根据解析规则提取相应信息,从而实现对 xls、xlsb、xlsm 和 slk 文件的

表 1 现有研究与本文工作对比

Table 1 Comparison between existing research and this paper

相关工作	支持文件格式	XLM 提取	反混淆	混淆常用函数模拟	家族识别
olevba	.xls、.slk	○	×	×	×
XLMMacroDeobfuscator	.xls、.xlsb、.xlsm	√	√	○(73)	×
SYMBEXCEL ^[1]	.xls、.xlsb、.xlsm	√	√	○(90/102)	×
XLMMRevealer	.xls、.xlsb、.xlsm、.slk	√	√	√(138)	√

注:“√”表示完全支持;“○”表示部分支持;“×”表示不支持。oletools 中的 olevba 脚本能够检测并提取出 slk 文件中的所有 XLM 公式,但只能提取 xls 文件中的部分公式,且不具有反混淆功能;XLMMacroDeobfuscator(XLMMMD)能够从 xls、xlsb、xlsm 这 3 种文件中提取 XLM 公式并反混淆;SYMBEXCEL 在 XLMMMD 的基础上结合符号执行的方法,能更好地应对利用执行环境信息进行条件分支判断类的混淆技术。

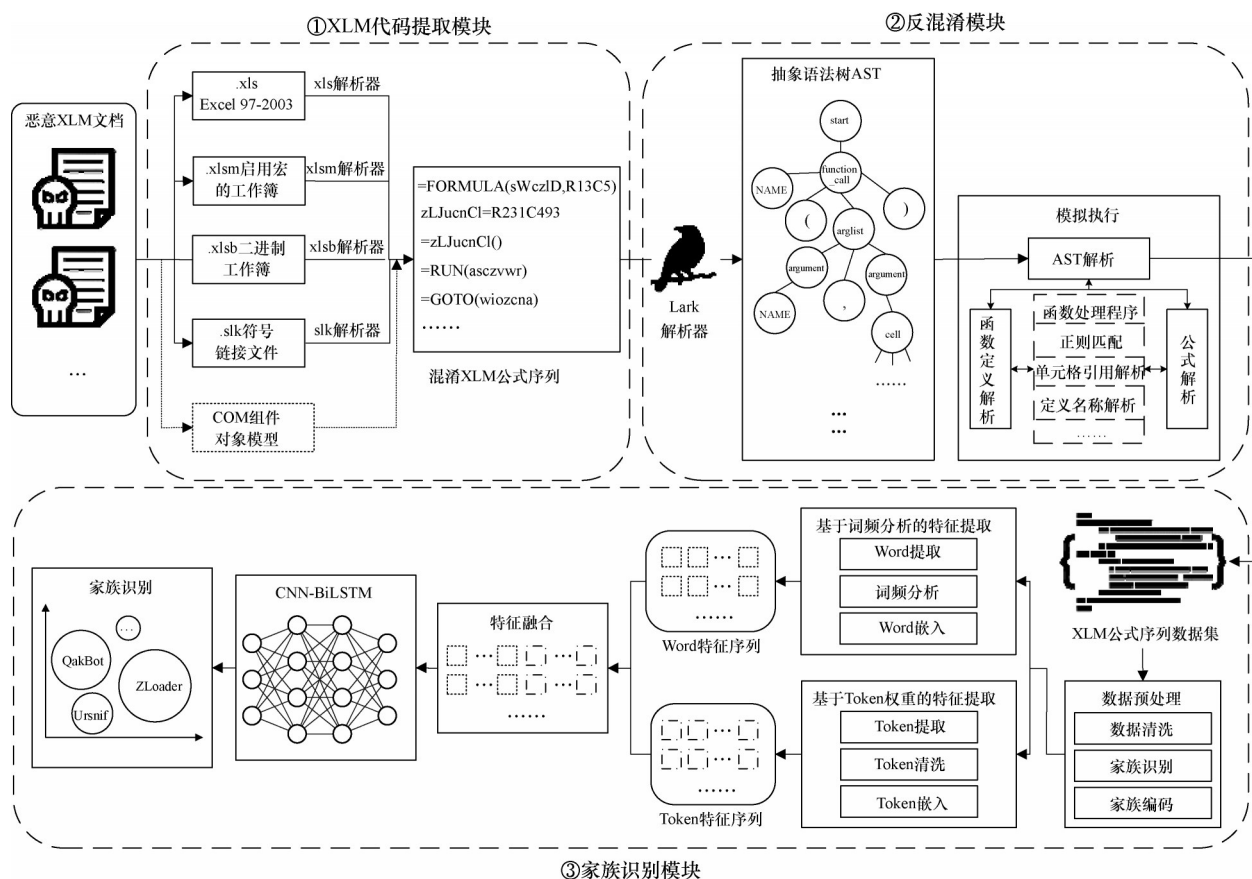


图2 XLMRevealer系统框架

Figure 2 System framework diagram of XLMRevealer

静态解析。

2) MS COM加载文件解析: COM是Microsoft 1993年提出的组件式软件平台,借助COM能与Excel对接,从而调用Excel内部解析器处理文件,解析情况与在Excel上真实打开的文档相同,且COM支持对所有Excel格式的加载。

实现静态文件解析和COM加载两种方式是必要的,具体原因如下:一方面,静态解析效率高、能够处理表内容被保护的情况,但实现与Excel相当的解析器以提取所有需要的表信息十分困难,且鲁棒性较差;另一方面,COM加载不受抗静态解析技术影响、能提取更多表信息,但要求具有MS Excel环境,无法读取被保护的单元格数据,且文件加载速度较慢。

综上所述,两种文件加载解析方法优势互补,能在保证解析效率的情况下充分应对攻击者采用的各种对抗文件解析的技术。

2.2 反混淆模块

2.2.1 反混淆模块构成

反混淆模块按照理论运行顺序逐条处理XLM公式,首先将其转换为抽象语法树,再调用模拟执行子模块解析AST,并对XLM公式进行去混淆,算法流程如算法1所示。

算法1 模拟执行流程

输入 $XL = XLWrapper(Path)$, XLM提取模块生成的Excel对象

输出 $F = [f_1, f_2, \dots, f_n]$ 其中, f_n 表示去混淆的第 n 条XLM公式

- 1) Initialization: $F \leftarrow \Phi, C \leftarrow \Phi, A \leftarrow \Phi, R \leftarrow \Phi$
- 2) $C = GetEntryPoint(XL)$ // 获取入口点单元格
- 3) $H = GetHandlerDict()$ // 获取模拟执行子模块字典
- 4) **WHILE**(C) **DO**
- 5) $A = AST(C.Formula)$ // 获取公

式对应的 AST

/* 根据 AST 类型调用对应的模拟执行子模块 */

6) **IF**(A.Type in H) **DO**

7) **R** = **H**[A.Type](A) // 模拟

执行获取结果

/* 根据具体情况作其他处理 */

8) **ELSE DO**

9) **R** = DeputyHandler (A)

10) **END**

11) **F.Append**(**R.Formula**) // 保存

反混淆结果

/* 按模拟执行顺序获取下一个包含公式的单元格 */

12) **C** = **GetNextCell**(C)

13) **END**

14) **RETURN F**

(1) 抽象语法树生成

XLMRevealer 从运行入口点开始, 按照执行顺序使用 Lark 工具逐个将公式解析成 AST。AST 是一种树状结构, 能分层抽象表示由形式化上下文无关语言编写的文本 (如 JavaScript、PowerShell 等代码) 的语义和语法特征^[14]。

在处理 XLM 公式时, 根据公式结构识别其具体类型。对于宏函数调用公式, 生成图 2 中的 `function_call` 子树及函数名节点, 并以递归下降的方式解析参数, 以处理参数中可能存在的函数嵌套调用。对于函数定义公式, 则将其解析为图 3 所示的 `assignment` 子树、函数名节点和单元格子树, 供后续模拟执行模块形成该自定义函数名与单元格地址键值对。

(2) 模拟执行

模拟执行模块抽象模拟 Excel 对宏代码的解析与执行, 从而实现反混淆。获得公式 AST 后, XLMRevealer 根据子树类型 (即公式类型) 调用相应的模拟执行子模块来模拟公式执行效果。

对于宏函数调用公式 AST, XLMRevealer 解析 `function_call` 子树获取调用的函数名, 再将参数子树列表传给对应的函数处理程序并进行函数执行模拟。函数处理程序集包含 138 个 Excel 4.0

所支持函数的模拟执行程序, 涵盖了数据集中观察到的所有函数及各种字符串、数值操作相关函数, 能够有效处理样本中宏函数的调用, 并应对各类利用字符串或数值进行混淆的方法。

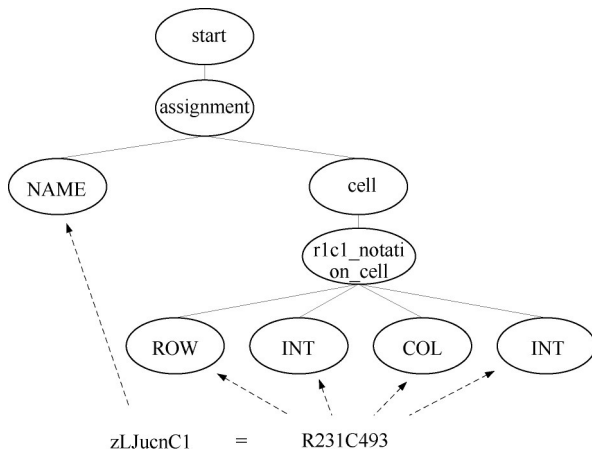


图 3 函数定义公式 AST 示例

Figure 3 Example of a function definition formula AST

总之, 模拟执行是反混淆成功的关键点, XLMRevealer 完善、健壮的模拟执行模块保证了反混淆高效、精确的执行。

2.2.2 反混淆关键技术点

XLMRevealer 优化反混淆的整体流程、完善对抗混淆手段的针对性算法, 以实现更高效准确的反混淆, 具体有以下关键技术点。

1) 根据样本文件格式、抗解析技术 (如修改文件结构相关二进制数据、保护表内容等)、混淆技术 (如利用大量执行环境信息等) 选择更适合的文件解析方法, 从而保证所获取 XLM 代码的正确性和完整性。

2) 完善抽象语法树生成模块以正确处理所有类型的公式, 有效对抗单元格内多条公式的混淆手段, 在处理该混淆时, XLMRevealer 将其解析成包含多个 `function_call` 子树的 `expression` 树, 并逐条传递给模拟执行模块处理。

3) 实现 138 个宏函数的模拟执行程序, 而现有工具最多仅实现 90 个。函数处理程序集不仅覆盖真实样本中观察到的所有常用函数, 还包括其他用于字符串与数值运算操作、执行恶意功能、消除恶意行为痕迹等的函数, 以及部分罕见但功能可能用于混淆的函数。丰富的函数处理程

序集确保模拟执行模块能正确解析宏函数调用公式。

4) 深入分析常用混淆手段, 完善模拟执行流程或设计针对性的反混淆算法。现有工具无法解析函数以及自定义与调用公式, 对自定义变量名、定义命名以及单元格引用的解析过于简单, 难以处理高级样本对这些常用混淆手段的复杂使用方式。本文通过实际测试确认上述技术的各种使用方式, 从而完善模拟执行流程。此外, 针对利用日期数据的混淆, XLMRevealer设计结合坏字符统计、单词编辑距离的算法以确认正确日期参数。

2.3 家族识别模块

在得到去混淆的XLM公式序列数据集后, 先对数据集进行预处理以获得高质量样本与家族类别标签, 然后通过特征提取、词嵌入、特征融合、模型构建及分类识别样本所属恶意家族。

2.3.1 数据预处理

为了更好地训练模型、提高识别准确率, 对得到的样本进行筛选。恶意XLM文档常常会利用隐藏宏表、调整字体颜色、在公式间插入大量空白单元格等方法躲避人工分析^[1], 因此在提取并反混淆XLM代码后再筛选样本更为合理。反混淆后得到的数据集中存在结果为空或未包含任一完整公式、家族样本数过少的样本, 本文去除这些样本后, 得到可用于后续家族分类的数据集。

随后, 需要标注样本数据集的家族标签, 文献^[1]认为在对恶意XLM文档样本标签的识别上MS Defender最为准确可靠, 因此本文使用MS Defender扫描样本数据集, 解析包含了检测结果的资源扫描与威胁检测日志MPLog, 提取家族标签, 如“TrojanDownloader:O97M/Dridex”对应“Dridex”家族。

2.3.2 特征提取

通常, 同一家族的样本在功能、关键函数调用顺序、自定义字符串等方面相似或相同, 因此提取的特征不仅要契合XLM语言本身的特点, 还要能体现不同样本在整体功能、代码结构、关键字字符串上的相似性。XLM本身语法简单, 代码以宏函数调用公式为主体, 夹杂变量声明公

式、函数自定义公式等, 主要通过宏函数的顺序来执行完成恶意功能。基于以上特点, 本文提出了Word(单词)级、Token(关键字, XLM中即各种函数名)级的特征提取方法。

(1) Word级的特征提取

本文通过将非英文字母字符替换为空格提取所有Words, 然后将字母转换为小写并过滤长度小于2的词, 获取各类函数名(宏函数、自定义函数等)、自定义变量名、Windows API注册名、重要参数字符串等强关联性特征, 这能有效体现同一家族样本相似性。

(2) Token级的特征提取

本文解析去混淆的XLM公式序列, 提取有序的XLM函数与方法关键字, 以反映样本在关键技术、代码功能上的相似性。

2.3.3 特征融合与词嵌入

本文基于词频排序、concat拼接等方法实现特征融合和词嵌入。具体而言, 本文统计训练集中每个Word和Token的频数, 分别从小到大排序并标上序号, Tokens的起始序号为Words尾序加1。然而, 测试集中常会出现训练集没有的新词, 即OOV(未登录词, out of vocabulary)问题, 本文通过将训练集中频数为1的Words和Tokens以及测试集中新出现的Words和Tokens分别记为“<UNK1>”“<UNK2>”以避免该问题^[15]。

随后, 将各样本的Words和Tokens特征序列对齐后采用concat方法拼接以实现Word和Token特征融合。各个词对应的序号会被嵌入形成向量, 从而生成一个高等级联合表征, 能够捕获细粒度特征, 并从不同维度挖掘家族样本关联性^[16]。

2.3.4 家族分类模型

XLMRevealer构建卷积神经网络(CNN, convolutional neural networks)和双向长短期记忆网络(BiLSTM, bidirectional long short-term memory)的组合模型来实现家族分类, 模型主要包含嵌入层、CNN层、BiLSTM层和全连接层(分类器)4个部分。

(1) 嵌入层

嵌入层接收先前拼接好的特征序列, 利用词

嵌入算法将各段特征序列中的所有序号转换为高维向量，从而生成样本特征向量。

(2) CNN 层

CNN 由卷积层和池化层构成，卷积层的卷积核起滤波器作用，能够有效学习文本序列局部特征。池化层采用最大池化对特征图采样，提取卷积结果中的高分特征并降维，使 XLMRevealer 可以学习到重要函数及关键字符串等特征。

(3) BiLSTM 层

LSTM 模型是一种特殊的 RNN，通过遗忘门、记忆门和输出门决定信息的取舍，并结合门的计算结果控制细胞状态，从而利用时间序列对输入进行分析并捕获较长距离的依赖关系。

BiLSTM 模型由前向 LSTM 和后向 LSTM 组成，能在特定时间范围内有效利用先前特征（通过前向状态）和未来特征（通过后向状态）^[17]。于是，由于 Token 特征中函数调用存在明显的时序关系，BiLSTM 层可以充分学习到 Token 语义特征、对应的恶意功能实现过程；而宏函数调用公式的函数名与参数、赋值公式的等号连接结构及左右参数等之间也存在着明显的关联关系，BiLSTM 层也能学习到各公式 Word 级语义特征。

(4) 全连接层（分类网络）

全连接层通常作为分类器设置在神经网络的最后一层。全连接层把特征矩阵展开成一维向量，从而将特征表示映射到样本标记空间，并传递给分类网络完成分类。XLMRevealer 使用 Softmax 层作为分类网络，实现恶意家族的识别。

3 实验与分析

本节分别针对 XLMRevealer 的反混淆模块和家族识别模块进行评估，设计三组实验验证两大功能的有效性，并在反混淆方面分析比较与现有工具（XLMMD、SYMBEXCEL）在性能上的差异。

3.1 实验数据与环境

3.1.1 实验数据

高质量的恶意 XLM 样本十分匮乏，更没有公认的基准数据集，于是本文从以下 5 个来源构建了包含 2 346 个样本的实验数据集：Lastline 网络安全公司公开数据集、Carbon Black 终端安全

公司的测试数据集、InQuest 安全公司 ZLoader 家族数据集、MalwareBazaar 恶意样本数据库、奇安信公司数据集。

3.1.2 实验环境

3 个工具在反混淆时都可能使用 COM 加载文件，而 SYMBEXCEL 需要运行在 Linux 中，为了使用 COM，SYMBEXCEL 还提供了 Windows 环境下的服务端代码为 Linux 提供 COM 功能。因此反混淆成功率测试实验中 XLMMD 和 XLMRevealer 在装有 Office 2019 的 Windows 10 物理机上运行，SYMBEXCEL 在 Linux 虚拟机中运行并使用 Windows 服务端提供的 COM 功能。

对于反混淆耗时比较测试，COM 加载耗时比静态解析高出一个数量级，使用 COM 会导致反混淆总体耗时差异不明显，因此该测试中均仅使用静态文件解析加载样本，3 个工具都在 Linux 虚拟机中运行。实验平台配置如表 2 所示。

表 2 实验平台配置

Table 2 Experimental platform configuration

关键硬件/软件	Windows 物理机	Linux 虚拟机
CPU	Intel i5-12500H 1×12 核	4×4 核
GPU	NVIDIA GeForce RTX 3050Ti	○
内存/显存	24 GB/16 GB	8G/○
CUDA	CUDA11	○
编程语言	Python 3.10	Python 3.10
深度学习库	PyTorch	○

注：“○”表示没有或不重要。

3.2 反混淆实验

从反混淆成功率、耗时与反混淆效果方面测试 XLMRevealer、SYMBEXCEL 和 XLMMD，并验证 XLMRevealer 的有效性和优越性。

3.2.1 反混淆成功率实验

为了评估 XLMRevealer 对真实世界样本的反混淆能力，本文在构建的数据集上进行了反混淆测试，3 个工具的综合反混淆成功率如图 4 所示，XLMRevealer 相比 XLMMD 和 SYMBEXCEL 分别提升了 20.8%、15.8%。

XLMRevealer 还能针对一些重要函数提示样本行为并提取关键 IOC，提取的部分 IOC 如图 5 所示。

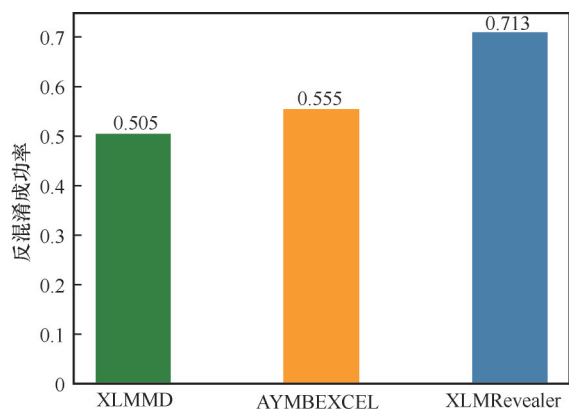


图4 3个工具的综合反混淆成功率

Figure 4 Comprehensive deobfuscation success rate of three tools

```

ALERT:
Alert Message: "Error! Sending report to Microsoft..."

EXEC:
Execute Command: "wmic.exe process call create 'mshta C:\ProgramData\mAuPRwjWflbuhQa.rtf'"

FWRITE:
Write File: "C:\Users\Public\b92fqwZR.reg"

```

图5 提取的部分IOC

Figure 5 Part of the extracted IOC

如何判断反混淆是否成功是没有统一标准的, 本文设计了多条规则分别统计反混淆成功率, 以从多个方面来衡量反混淆结果的有效性。

1) 反混淆结果非空。样本的抗解析、抗反混淆技术可能使反混淆代码运行出现异常终止, 导致没有存入结果或结果为空。

2) 反混淆结果行数在十到十万之间。异常情况可能导致反混淆提前结束, 对循环公式反混淆失败, 则可能进入无限循环而出现过多结果。

3) 反混淆结果至少包含一个重要函数。恶意样本常会对关键函数进行混淆, 文献[1]选取用于执行恶意功能的“EXEC”“REGISTER”“CALL”函数, 本规则在此基础上进行扩充, 还选取用于消除恶意行为痕迹的下列函数: “CLOSE”“UNREGISTER”“FILE. DELETE”“FILE.CLOSE”“WORKBOOK.HIDE”。

4) 反混淆结果中至少包含一个重要特征字符串。重要函数调用的参数是否正确, 极大程度上反映了反混淆成功与否, 根据真实样本调用3)中重要函数时的常见参数, 设置了以下特征字符串集合: “exe”“dll”“powershell”“ps1”“html”“lnk”“bat”“mshta”“rtf”“URL”“Kernel32”

“Shell32”“Download”“http”“C:\”“vbs”。

本文统计3种工具在各条规则以及同时满足规则3)、4)下的反混淆成功率, 如表3所示, 在规则1)、2)下, XLMRevealer有极高的反混淆成功率, 证明其对于绝大部分真实样本都具有较强的反混淆能力。在规则3)、4)以及同时满足这两条规则的条件下, XLMRevealer的成功率相较XLMMMD和SYMBEXCEL分别提升了20.0%和12.1%、11.6%和6.5%以及18.3%和15.8%。此外, 本文发现在规则3)、4)下反混淆成功率比在规则1)、2)下降了不少, 究其原因, 一方面是在存在未能完全反混淆的样本, 另一方面则是观察到大量样本本身不作为恶意负载加载器使用, 所以并不会调用上述重要函数。

表3 不同规则下3种工具的反混淆成功率

Table 3 Deobfuscation success rate of tools under different rules

规则	XLMMMD	SYMBEXCEL	XLMRevealer
1)结果非空	0.675	0.748	0.946
2)行数在十到十万之间	0.569	0.590	0.839
3)包含重要函数	0.421	0.500	0.621
4)包含重要字符串	0.466	0.517	0.582
5)重要函数及字符串	0.393	0.418	0.576

3.2.2 反混淆效率实验

本文在Linux虚拟机中运行3个工具并记录反混淆每个样本的耗时, 处理各样本的用时上限设置为180s。最后, 以20为步长按序抽样并绘制了反混淆效率散点图, 反混淆耗时如图6所示。

总体而言, XLMRevealer的反混淆高效且稳定, 平均用时为0.512s, 远低于其他工具。此外, 耗时较高的几个样本使用了条件循环或高循环次数写文件, XLMRevealer在处理这些循环上表现良好, XLMMMD和SYMBEXCEL则均多次出现进入无限循环而达到用时上限的情况。

SYMBEXCEL利用符号执行的方法判断函数返回值及分支循环条件, 其平均耗时高出了一个数量级。由于其模拟执行逻辑不够完善、对单

元格引用解析存在较大缺陷、仅覆盖了少量反混淆技术等,其总体反混淆成功率不够高且在面对大量样本时效率很低。

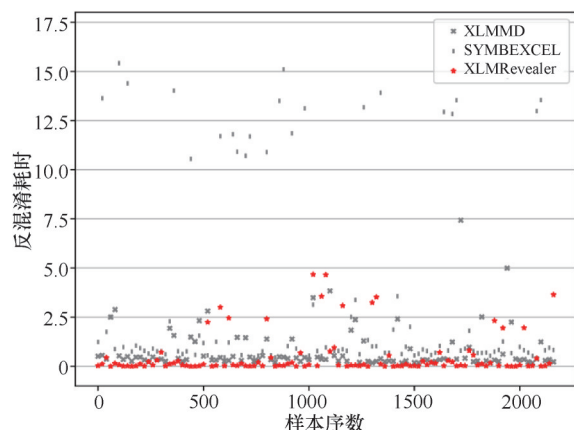


图6 3个工具的反混淆耗时

Figure 6 Deobfuscation time of three tools

3.2.3 反混淆能力分析

为了直观比较各工具对常见混淆技术的反混淆效果,使用这3种工具处理图7(a)所示案例。

该案例中的宏表名为“test”,R1C1单元格存放两条公式,其中“SET.VALUE”将“MID”得到的字符串“1C”存入R5C4;“DEFINE.NAME”为地址引用参数指向的R1C2单元格设置定义名称“asd”;R2C1中的赋值公式声明一个名为“sdwCWvq”的函数,并通过定义名称“asd”指向R1C2,然后再调用该函数,从而跳转到R1C2,并向下运行,直到由“RETURN”返回调用点,最终由“HALT”结束宏运行。

总的来说,该案例使用了单元格多公式、字符串混淆、多种形式的单元格引用、赋值公式自定义函数、控制流跳转等混淆方法。3种工具的反混淆结果如图7(b)~图7(d)所示,结果表明,前两种工具难以处理该案例。

具体而言,XLMMD无法处理赋值公式以及单元格内多公式的混淆问题。SYMBEXCEL能解析单元格内的各条公式,但仅在反混淆结果中显示最后一条公式,而由于其单元格地址解析功能不够完善,没有正确处理单元格引用参数,仅打印参数字符串;此外,其还缺乏对函数自定义与调用的解析,这些原因导致模拟执行最终没有跳转到R1C2单元格,无法实现完整的反混淆。

与之相比,本文提出的XLMRevealer具有更加完善且鲁棒的单元格地址引用解析算法,支持各种混淆技术的解析处理,且反混淆结果清晰明确。

```
[R1C1]=SET.VALUE(D5,MID("dwc1C8W3",4,2))
=DEFINE.NAME("asd","=R"&R[4]C[3]&CHAR(50))
[R2C1]sdwCWvq=asd
[R3C1]=sdwCWvq()
[R4C1]=HALT()

[R1C2]"B2"
[R2C2]=ALERT(1)
[R3C2]=RETURN()
```

(a) 混淆 XLM 公式

```
False
False
=sdwCWvq()
=HALT()
```

(b) XLMMD 反混淆结果

```
DEFINE.NAME(asd,"=R1C2")
sdwCWvq=asd
=sdwCWvq()
=HALT()
```

(c) SYMBEXCEL 反混淆结果

```
SET.VALUE(test!$D$5,"1C")
DEFINE.NAME("asd",test!$B$1)
sdwCWvq=test!$B$1
sdwCWvq()->GOTO(test!$B$1)
ALERT(1)
RETURN()
HALT()
```

(d) XLMRevealer 反混淆结果

图7 混淆 XLM 公式和 3 种工具的反混淆结果

Figure 7 The obfuscated XLM formulas and deobfuscation results of three tools

3.3 家族分类实验

本文根据 XLM 代码特点构建了 CNN 和 BiLSTM 的组合模型,为验证该模型实现家族分类的有效性,选取数据集中反混淆结果非空的家族样本,从反混淆结果中提取 Word 和 Token 特征,并输入模型进行分类。此外,还选取其他基线模型进行实验,以比较 XLMRevealer 与现有模型在处理 XLM 家族分类问题上的差异。

3.3.1 实验设置

(1) 分类实验家族数据集

为验证所提取特征的有效性以及分类模型的鲁棒性,从先前构建的数据集中提取出满足反混淆成功判定规则 1) (反混淆结果非空) 的 2 220 个

样本用于分类实验。本文利用 MS Defender 扫描样本集, 解析其生成的资源扫描与威胁检测日志 MPLog, 以获取各个样本的家族标签。

2 220 个样本中有 2 000 个被打上了标签, 共包含 27 个不同的家族, 然而绝大部分家族的样本数量极少, 甚至仅有一两个, 不足以用于分类实验。因此本文最后仅能选取 8 个家族: Donoff、Dridex、EncDoc、Gozi、Mailcab、ObfBook、QakBot、ZLoader, 构建了如表 4 所示的共 1 961 个样本的家族分类实验数据集, 每个家族的样本按照 6:2:2 的比例划分训练集、验证集和测试集。

表 4 家族分类实验数据集

Table 4 Dataset of family classification experiment

类别	恶意 XLM 家族	总数	训练集	验证集	测试集
0	Donoff	15	9	3	3
1	Dridex	257	155	51	51
2	EncDoc	1025	615	205	205
3	Gozi	11	7	2	2
4	Mailcab	26	16	5	5
5	ObfBook	32	20	6	6
6	QakBot	95	57	19	19
7	ZLoader	500	300	100	100

(2) 评估指标

为较全面地评价 XLMRevealer 对恶意 XLM 样本进行家族分类的能力, 本文使用分类任务中的常见评估指标: 精确率 (Precision)、召回率 (Recall)、 F_1 值 (F_1 -score) 和准确率 (Accuracy) 评估实验结果。

恶意 XLM 家族分类属于多分类任务, 而上述实验数据集中各家族样本数量分布很不均衡, 为更准确地评价 XLMRevealer 家族分类效果, 本文使用加权计算后的精确率、召回率、 F_1 值和准确率, 其中, 各家族评估指标的权重为家族样本数量占总样本数量的比例。

(3) 对比实验环境与模型

家族分类实验在表 2 所示的 Windows 物理机上进行, 为验证 XLMRevealer 所使用的模型在恶意 XLM 家族分类问题上是否具有更好的效果, 构建了两种基线 (baseline) 模型进行对比实验。

1) 机器学习基线模型。使用经典机器学习算法构建基线模型, 包括支持向量机 (SVM, support vector machine)、随机森林 (RF, random forest) 和逻辑回归 (LR, logistic regression) 3 个模型。

2) 深度学习基线模型。利用 PyTorch 构建该类基线模型, 包括 CNN、LSTM、BiLSTM 和门控循环单元 (GRU, gated recurrent unit) 4 个模型。

3.3.2 实验结果

XLMRevealer 所使用的 CNN-BiLSTM 模型与机器学习、深度学习各基线模型的实验结果如表 5 所示。由表 5 可知, XLMRevealer 在精确率、召回率、 F_1 值和准确率上均取得了最佳值。

表 5 各模型实验结果

Table 5 Experimental results for each model

模型	精确率	召回率	F_1 值	准确率
LR	0.885 1	0.887 5	0.880 7	0.887 5
SVM	0.903 3	0.910 5	0.905 0	0.910 5
RF	0.912 2	0.928 4	0.915 7	0.928 4
CNN	0.923 0	0.902 8	0.905 0	0.902 8
LSTM	0.864 3	0.882 4	0.862 5	0.882 4
BiLSTM	0.867 9	0.887 5	0.867 5	0.887 5
GRU	0.936 8	0.920 7	0.926 3	0.920 7
XLMRevealer	0.942 3	0.948 8	0.939 5	0.948 8

实验结果验证了 XLMRevealer 能够有效识别恶意 XLM 文档的代码特征与主要功能, 以实现准确的恶意家族分类。CNN-BiLSTM 组合模型有效捕获各公式中宏函数与参数的局部文本序列特征, 以及整个文档所调用的所有宏函数间长距离的依赖关系和语义知识, 具有更好的分类效果。

3.4 混淆对家族分类的影响实验

为探索反混淆对家族分类的影响, 本文在提取样本原始 XLM 代码而不进行反混淆的情况下完成家族分类实验。

然而, 考虑各家族样本使用的混淆技术类也可能存在一定差别, 此时模型将识别混淆技术的特征, 导致分类准确率下降不明显。于是本文使用统一的混淆方法对反混淆后的代码再混淆并进行分类实验。混淆方法为利用 “CHAR”

“CONCATENATE”进行字符串混淆，然后使用“FORMULA.FILL”实现数据流混淆，最后调用“RUN”执行存储在目标单元格的原始代码，混淆后效果如图8所示。

原始代码:	[R1C1]=calc="calc.exe" [R2C1]=EXEC(calc) [R3C1]=HALT()
混淆后代码:	[R1C1]=FORMULA.FILL(CONCATENATE(CHAR(99), CHAR(97),CHAR(108),CHAR(99),CHAR(61), CHAR(34),CHAR(99),CHAR(97),CHAR(108), CHAR(99),CHAR(46),CHAR(101),CHAR(120), CHAR(101),CHAR(34)),D1) [R2C1]=FORMULA.FILL(CONCATENATE(CHAR(61), CHAR(69),CHAR(88),CHAR(69),CHAR(67), CHAR(40),CHAR(99),CHAR(97),CHAR(108), CHAR(99),CHAR(41)),D2) [R3C1]=FORMULA.FILL(CONCATENATE(CHAR(61), CHAR(72),CHAR(65),CHAR(76),CHAR(84), CHAR(40),CHAR(41)),D3) [R4C1]=RUN(D1) [R5C1]=HALT()

图8 原始代码与统一混淆后代码示例

Figure 8 Example of original code and uniformly obfuscated code

对反混淆前与统一混淆后 XLM 代码的家族分类准确率如图9所示。

由图9可知，对于未去混淆的 XLM 代码，其他所有基线模型的分类准确率平均值为 85.46%，XLMRevealer 的家族分类准确率为 89.58%，高出了 4.12%。而对比 3.3 节中的实验

结果，所有模型的准确率仅有小幅度降低，说明即使无法很好学习到代码功能和语义特征，模型仍能通过识别混淆技术的特征而进行较为准确的分类；而对于反混淆后再统一混淆的代码，各家族样本失去了混淆技术上的特征，包含的大量仅用于混淆的无意义数据和代码又使分类模型无法学习到样本关键特征、语义关系等，导致各深度学习模型分类准确率低至接近 50%。

上述实验结果不仅说明分类模型能学习真实样本中各家族使用的混淆技术特征，还验证了反混淆对家族分类具有极大的促进作用，在所有样本使用相同的混淆技术时，分类模型几乎失效。

4 结束语

基于恶意 XLM 文档的钓鱼攻击在近年来爆发，为解决安全人员难以分析重混淆 XLM 文档的真实功能并获取 IOC 的问题，以及满足现实世界中检测与分析海量恶意样本时的家族分类需求，本文实现了 Excel 4.0 宏自动化反混淆与家族分类系统 XLMRevealer。

XLMRevealer 能够有效处理近 10 种 XLM 混淆手段，反混淆结果可读性更好，能够自动提取

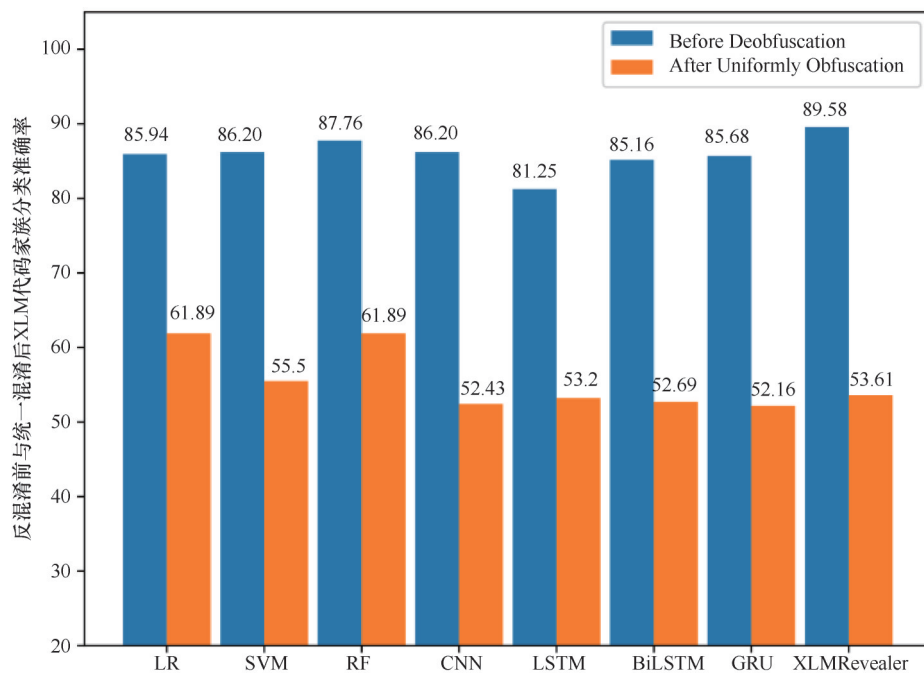


图9 反混淆前与统一混淆后 XLM 代码家族分类准确率

Figure 9 Family classification accuracy of XLM codes before deobfuscation and after uniformly obfuscation

重要IOC,在构建的包含2 346个真实样本的数据集上,综合反混淆成功率相比XLMM和SYMBEXCEL有较大的提升,反混淆效率高。

XLMMRevealer提取Word与Token特征,构建了CNN和BiLSTM组合模型,分类效果优于基线模型。本文研究XLM代码关键特征,根据代码特点构建分类模型,并进行对比实验以验证模型有效性,以满足现实世界对海量样本的分析溯源需求。此外,本文通过对恶意样本未反混淆的原始代码、统一混淆后的代码进行分类实验,证明模型能够识别各家族混淆技术的特征,验证了反混淆对家族分类的促进作用。

后续工作将重点分析样本中出现的特殊混淆手段,提出针对性反混淆方法,以提升反混淆成功率;还可探索反混淆程度评价方法,揭示混淆前后代码的变化特征;还将根据XLM代码特点继续打磨提取的特征,构建效果更好的家族分类模型,在更多家族的大数据集上实现准确分类。

此外,现有反混淆研究与XLMMRevealer均针对已有混淆技术设计对应算法,但在面对新型未知混淆技术时,XLMMRevealer只能依赖其较为完善的反混淆逻辑与框架,具体反混淆效果有待验证。基于反混淆领域研究现状,想要能够准确处理已知和未知混淆技术,可能需要将目光投向大语言模型(LLM, large language model)。

LLM技术在近年兴起,基于大尺寸模型和大规模高质量数据等条件因素,LLM不仅具有极强的自然语言文本的理解与生成能力,还能够处理绝大多数编程语言问题。LLM已在代码理解和代码生成方面展现出卓越的性能,而对代码的反混淆本质上就是理解混淆代码并生成去混淆代码。因此,通过对混淆样本和原始样本的分析学习,LLM应能识别样本是否经过混淆,而基于语法规则解析和上下文分析能力,LLM可以实现高效且准确的反混淆。后续工作将尝试探索LLM在XLM反混淆上的应用。

参考文献:

- [1] RUARO N, PAGANI F, ORTOLANI S, et al. SYMBEXCEL: automated analysis and understanding of malicious Excel 4.0 macros[C]//Proceedings of the 2022 IEEE Symposium on Security and Privacy (SP). Piscataway: IEEE Press, 2022: 1066-1081.
- [2] YAN J, WAN M, JIA X K, et al. DitDetector: bimodal learning based on deceptive image and text for macro malware detection[C]//Proceedings of the 38th Annual Computer Security Applications Conference. New York: ACM, 2022: 227-239.
- [3] MIURA H, MIMURA M, TANAKA H. Macros finder: do you remember LOVELETTER? [C]//International Conference on Information Security Practice and Experience. Cham: Springer, 2018: 3-18.
- [4] KIM S, HONG S, OH J, et al. Obfuscated VBA macro detection using machine learning[C]//Proceedings of the 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). Piscataway: IEEE Press, 2018: 490-501.
- [5] MIMURA M, OHMINAMI T. Towards efficient detection of malicious VBA macros with LSI[C]//International Workshop on Security. Cham: Springer, 2019: 168-185.
- [6] MIMURA M, MIURA H. Detecting unseen malicious VBA macros with NLP techniques[J]. Journal of Information Processing, 2019, 27: 555-563.
- [7] LAIN D, KOSTIAINEN K, ČAPKUN S. Phishing in organizations: findings from a large-scale and long-term study[C]//Proceedings of the 2022 IEEE Symposium on Security and Privacy (SP). Piscataway: IEEE Press, 2022: 842-859.
- [8] 宋文纳, 彭国军, 傅建明, 等. 恶意代码演化与溯源技术研究[J]. 软件学报, 2019, 30(8): 2229-2267.
- [9] SONG W N, PENG G J, FU J M, et al. Research on malicious code evolution and traceability technology[J]. Journal of Software, 2019, 30(8): 2229-2267.
- [10] LE Q V, MIKOLOV T, JIA H F, et al. Distributed representations of sentences and documents[EB]. arXiv Preprint arXiv: 1405.4053. 2014.
- [11] HENDLER D, KELS S, RUBIN A. Detecting malicious PowerShell commands using deep neural networks[C]//Proceedings of the 2018 on Asia Conference on Computer and Communications Security. New York: ACM, 2018: 187-197.
- [12] HENDLER D, KELS S, RUBIN A. AMSI-based detection of malicious PowerShell code using contextual embeddings[C]//Proceedings of the 15th ACM Asia Conference on Computer and Communications Security. New York: ACM, 2020: 679-693.
- [13] MIKOLOV T, CHEN K, CORRADO G, et al. Efficient estimation of word representations in vector space[EB]. arXiv Preprint arXiv: 1301.3781. 2013.
- [14] TAJIRI Y, MIMURA M. Detection of malicious PowerShell using word-level language models[C]//AOKI K, KANAOKA A. International Workshop on Security. Cham: Springer, 2020: 39-56.
- [15] AL-TAHARWA I A, LEE H M, JENG A B, et al. JSOD: JavaScript obfuscation detector[J]. Security and Communication Networks, 2015, 8(6): 1092-1107.
- [16] LUONG T, SUTSKEVER I, LE Q, et al. Addressing the rare word problem in neural machine translation[C]//Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). Stroudsburg, PA, USA: Association for Computational Linguistics, 2015: 1112-1121.

ciation for Computational Linguistics, 2015: 11-19.

- [16] YANG X Z, PENG G J, ZHANG D N, et al. PowerDetector: Malicious PowerShell script family classification based on multi-modal semantic fusion and deep learning[J]. China Communications, 2023, 20(11): 202-224.

- [17] HUANG Z H, XU W, YU K. Bidirectional LSTM-CRF models for sequence tagging[J]. arXiv Preprint arXiv: 1508.01991. 2015

[作者简介]



李晨光 (1999-), 男, 湖北十堰人, 武汉大学硕士生, 主要研究方向为恶意代码分析。



杨秀璋 (1991-), 男, 贵州施秉人, 武汉大学博士生, 主要研究方向为恶意代码分析。



彭国军 (1979-), 男, 湖北荆州人, 武汉大学教授、博士生导师, 主要研究方向为网络安全、信息系统安全。