文章编号:1671-4598(2021)11-0171-05 DOI:10.16526/j.cnki.11—4762/tp.2021.11.031 中图分类号:TP309 文献标识码:A

基于图变换的虚拟机保护增强方法

谢 鑫,罗莉霞,陈 敏
(湖南信息学院 计算机科学与工程学院,长沙 410151)

摘要:针对虚拟机框架不同模块连接之间的强固定性,提出一种基于图变换的虚拟机保护增强方法;首先将虚拟机结构框图转换为有向完全图,然后运用多重等价变换策略对完全图中节点模块进行等价变形,最后运用多样化虚拟机对不同节点模块进行嵌套保护;采用了混沌不透明谓词,跳转表和指令等价规则等关键技术实现了原型系统,通过实验验证了系统的可行性和方法的有效性。

关键词:图变换;代码数据跳变;虚拟机保护;混沌不透明谓词;等价指令替换

Virtual Machine Protection Enhancement Method Based on Graph Transformation

XIE Xin, LUO Lixia, CHEN Min
(School of Computer Science and Engineering, Hunan Institute of Information Technology, Changsha 410151, China)

Abstract: For the strong fixity between the connections of different modules of the virtual machine, a virtual machine protection enhancement method based on graph transformation was proposed. Firstly, the structure graph of virtual machine framework was converted into a directed complete graph, then multiple equivalence transformation strategies were applied to transform the nodes in the complete graph, and finally diverse virtual machines were applied to different node modules for nest protection. The prototype system for virtualization code was implemented by using the key techniques such as chaotic opaque predicates, jump tables and instruction equivalence rules. Experimental results show that the system is feasible and the method is effective.

Keywords: graph transformation; code and data transformation; virtual machine protection; chaotic opaque predicates; equivalent instruction substitution

0 引言

伴随着全球通信产业链的不断升级,通信软件在不断革新和优化的通信产业生态环境中,所处的地位变得越来越重要。但随之而来的是,对于通信软件的侵权及其漏洞的攻击,也变得越发的频繁。恶意攻击者通过采用静态和动态的软件逆向分析方法和工具,探查通信软件内部数据和代码,明晰软件内部的安全保护机制,通过修改和控制通信软件的内部逻辑,绕过内置安全保护机制,实现对于关键数据,算法和功能模块的获取。

目前对于通信软件的保护研究聚焦在于采用代码混淆技术,在保证通信软件功能不变的前提下,对指令进行变形和膨胀,对逻辑和结构进行复杂和模糊,或采用加密技术对关键代码和数据进行保护。而与代码混淆和代码加密不同的是,采用代码虚拟化技术对通信软件进行保护,既不是在原始代码层面进行功能等价变换,也不是提前将重要代码或数据进行加密,需要执行时再在内存中进行解密。而是将目标代码翻译成攻击者不熟悉的其他类型中间语言,在原指令系统中又叠加了另一套指令系统。攻击者要想正确理解目标代码,需要分析目标代码所对应的指令系统框架和新叠加的虚拟指令系统。

针对单虚拟机框架内部模块安全的研究工作如下。虚拟指令解释函数(Handler)模块:采用模拟退火算法随机化对 Handlers 进行指令乱序[1];通过等价指令替换和切分乱序对 Handlers 进行多样化[2];对 Handlers 序列进行动态加解密[3];基于时间多样性设计对 Handlers 进行多样化[4];采用数据混淆引擎对 Handlers 进行变形[5]。虚拟指令调度器(Dispatcher)模块:基于控制流迭代混淆和随机切分加密方法的 Dispatcher 安全性增强[6];防止攻击者对虚拟机进行定位的 Dispatcher 模块隐藏[7];增加程序多样性的多 Dispatchers 程序控制单元插入[8]。虚拟机指令(Bytecode)模块:采用虚拟花指令序列与虚拟指令模糊变换技术改进虚拟机指令模块强度[9];对虚拟指令字节码进行加解密[10]。虚拟机上下文(VMContext)模块:基于多组寄存器值变形的虚拟机上下文复杂化[11]。

针对单虚拟机框架模块连接安全的研究工作有:打乱字节码指令的操作码和处理程序之间的对应关系的虚拟指令随机化[12];减少静态虚拟指令内存暴露的基于多样化调度器的虚拟代码折叠[13]。

针对虚拟机整体框架安全的研究工作有:采用多虚拟机框架混淆从虚拟机操作码到本地机器指令的映射[14];采

收稿日期:2021-03-30; **修回日期**:2021-05-10。
基金项目:湖南省自然科学基金项目(2019JJ50413;2019JJ70075);湖南省教育厅科学研究重点项目(18A512;20A351)。
作者简介:谢 鑫(1986-),男,博士,湖南邵阳人,副教授,CCF 会员,主要从事软件安全、智能医疗影像方向的研究。
引用格式:谢 鑫,罗莉霞,陈 敏. 基于图变换的虚拟机保护增强方法[J]. 计算机测量与控制,2021,29(11):171-175.

用多套虚拟机环境随机选择和执行所构造的混淆基本块和关键代码[15]；基于不同强度的多重虚拟机嵌套保护框架[16]；基于多虚拟机的核心代码并行化保护框架[17]。

上述研究工作从不同角度提升了虚拟机保护技术的安全性，但从虚拟机整体框架看，不同模块间的连接结构具有较强的固定性，攻击者依旧较容易实现：先从高维视角整体理解虚拟机保护框架，然后再进一步分析各模块内部功能，从而实现保护机制的破解。本文针对虚拟机模块连接的固定结构，提出一种基于图变换的代码虚拟化安全保护方法：首先对虚拟机基本结构图进行变形，然后采用花指令嵌入、等价指令变换和代码数据跳变方法对不同模块内部指令序列进行变形，生成多样化虚拟机集，最后对虚拟机不同模块进行嵌套变形。

1 基于图变形的虚拟机保护框架增强

攻击者对受虚拟机保护的代码分析流程为：先明晰虚拟机保护整体结构和解释执行框架，再深入对各模块功能进行分析。为了增加虚拟机模块间连接结构和模块内部代码的复杂性，提升虚拟机保护强度，基于图的基本操作，对虚拟机的框架结构进行混淆，提出一种基于图变换的虚拟机框架变形方法，如图 1。

基本思想为：首先构建虚拟机结构图如图 1（a），然后通过增加不透明谓词和逻辑分支将其转化为正则图如图 1（b），再运用花指令嵌入、等价指令替换和代码数据跳变方法，对正则图节点中部分指令序列进行等价变形如图 1（c），最后基于宽度和深度嵌套策略，对部分节点中的核心指令序列进行多虚拟机保护如图 1（d）。

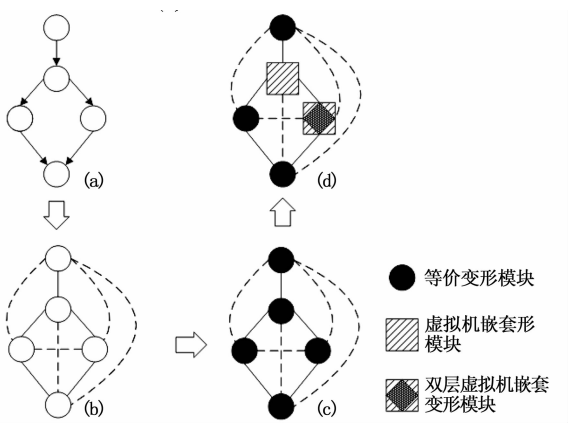

图 1 基于图变换的虚拟机混淆

1.1 相关定义

首先给出虚拟机结构图，虚拟机结构图混淆等相关定义如下。

定义 1（虚拟机 VM）：在虚拟机框架 VM 中，$VM=\{BC, HD, DP, JT, VMC, VMS, VME\}$，$BC$ 表示字节码，HD 表示虚拟指令解释函数，DP 表示虚拟指令调度器，JT 表示跳转表，VMC 表示虚拟机上下文，VMS 表示虚拟机入口代码，VME 表示虚拟机出口代码。

定义 2（虚拟机结构图 VM_G）：由虚拟机框架 VM 各模块构成的结构图 VM_G，$VM=\{V, E\}$，V 表示虚拟机 VM 所有模块集合，E 表示模块间构成的控制流关系集合，若结构图不含平行边和环，则称有向简单虚拟机结构图。

定义 3（虚拟机结构图混淆 $OBVM_G$）：设 VM 结构构成 n 阶有向简单图，C 为 VM 中的核心代码，$OBVM_G(VM)=f_1(f_2, f_1$ 为将有向简单虚拟机结构图 VM_G 转换为语义等价且每个顶点都邻接到其余 $m-1$ 个顶点上的 m 阶有向完全虚拟机结构图的变换算法，f_2 为一种将 C 变换为语义等价的混淆代码 C_O 的代码混淆算法。

定义 4（多样化虚拟机集 $OBVMS$）：用 $OBVM_G$ 对 VM 进行混淆，生成多样化虚拟机集 $OBVMS=\{VM_1, VM_2, VM_3, \cdots, VM_n\}$，对于 $(i, j(n, VM_i=VM_j$，表示虚拟机 VM_i 和 VM_j，表示虚拟机 VM_i 和 VM_j 功能等价，$|VMS|$ 表示多样化虚拟机的数量。

1.2 虚拟机框架混淆方法

基于图变换的虚拟机框架变形原理，对虚拟机框架 VM_G 进行混淆分三步进行。首先采用虚拟机结构图变换算法对虚拟机框架进行变形，再基于混淆策略生成多样化虚拟机框架集 $OBVMS$，最后进行嵌套保护构建 $GOB-VM$ 保护框架，具体如下。

1）虚拟机结构图变换。

步骤 1：基于虚拟机 VM 保护代码构建其结构图 VM_G；

步骤 2：对 $VM=\{BC, HD, DP, JT, VMC, VMS, VME\}$ 代码模块进行切分，每一模块切分后子模块数目分别记为 $\{k_{BC}, k_{HD}, k_{DP}, k_{JT}, k_{VMC}, k_{VMS}, k_{VME}\}$；

步骤 3：构建切分后的虚拟机 VM 结构图 $VM_{G'}=\{V, E\}$；

步骤 4：将 $VM_{G'}$ 转化为 $k=k_{BC}+k_{HD}+k_{DP}+k_{JT}+k_{VMC}+k_{VMS}+k_{VME}$ 阶有向完全虚拟机结构图。

2）多样化虚拟机框架集生成。

花指令嵌入策略：将对程序语义不产生影响的花指令，嵌入到目标指令序列上下文中，花指令和目标指令之间不产生依赖关系，嵌入前后的目标指令序列语义不发生改变，如图 2（a）所示。

等价指令替换策略：采用等价指令模板对目标指令序列进行替换，目标指令经过混淆后不出现在混淆指令序列之中，替换后的指令片段相互之间具有依赖关系，如图 2（b）所示。

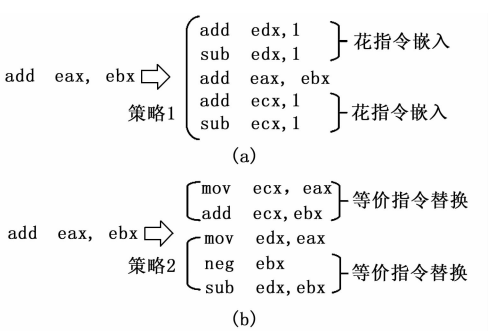

图 2 花指令嵌入和等价指令替换

代码数据跳变的基本过程，如图 3 所示。

步骤 1：设原始指令序列为（c_1，c_2，c_3，…，c_k），将其表示为十六进制数据序列，记为 D；

步骤 2：将 D 进行随机切分，切分后的序列记为（d_1，d_2，d_3，…，d_l）；

步骤 3：将（d_1，d_2，d_3，…，d_l）转化为能输出对应数据序列的 *Mealy* 自动机代码，记为（M_1，M_2，M_3，…，M_l）；

步骤 4：对（M_1，M_2，M_3，…，M_l）进行（m_1，m_2，m_3，…，m_l）次迭代膨胀变形[18]，转化为（M_1'，M_2'，M_3'，…，M_l'）对应的代码模块；

步骤 5：对（M_1'，M_2'，M_3'，…，M_l'）代码模块进行拼接，使其能输出数据序列 D，并最终转化为原始指令序列（c_1，c_2，c_3，…，c_k）。

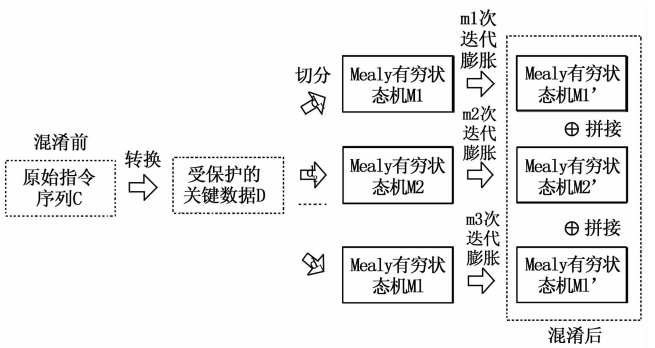

图 3　基于 Mealy 机膨胀的代码数据跳变

3）虚拟机嵌套。

采用宽度或深度嵌套策略如图 4，对多样化虚拟机集合中的单虚拟机核心代码模块进行多虚拟机保护。

宽度嵌套策略：针对单虚拟机 VM_1 中的所有模块，随机选择部分核心指令序列，采用生成的多样化虚拟机集 $\{VM_2, VM_3, \cdots, VM_d\}$ 对其进行保护。

深度嵌套策略：针对单虚拟机 VM_1 中的单个模块，随机选择部分核心指令序列，采用虚拟机 VM_2 对其进行保护，生成虚拟机 VM_1' 模块然后再采用虚拟机 VM_3 对 VM_1' 相同模块进行保护，生成 VM_1''，依次类推。

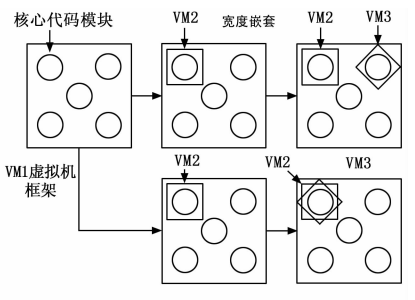

图 4　虚拟机框架核心代码嵌套

2　系统关键技术

2.1　基于不透明谓词的结构图连接

考虑到让混沌系统具有更不确定的属性和更复杂的相空间，选用二次映射混沌系统来构造安全性更高的不透明谓词集合 $\{OP_1, OP_2, \cdots, OP_n\}$，然后再从该集合中随机选取不透明谓词对虚拟机框架模块进行连接，实现 m 阶有向简单虚拟机结构图向 m 阶有向完全虚拟机结构图的变换。

若该图中 m 个节点所对应的出度，分别为 $\{o_1, o_2, \cdots, o_m\}$，则在每个节点和下个节点连接中，添加 $\{m-o_1-1, m-o_2-1, \cdots, m-o_m-1\}$ 个不透明谓词，增加 $\{2(m-o_1-1), 2(m-o_2-1), \cdots, 2(m-o_m-1)\}$ 条控制连接边，如图 5 所示。

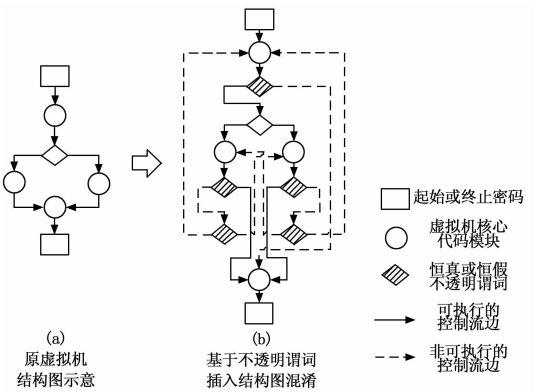

图 5　基于跳转表的结构图连接

2.2　基于跳转表的结构图混淆

将基于不透明谓词的虚拟机混淆结构图投射到内存空间，在每一个不透明谓词结束的地方，都会有类如 $jmp\ b_1$ 无条件或 $jnz\ b_2$，…，$jg\ b_n$ 等有条件等指令；然后针对虚拟机变换后模块中所有跳转指令，构建统一的跳转表，采用统一的 $call\ f$ 对其进行控制流转移，如图 6 所示。若想进一步增强跳转表的安全性，可以将其嵌入到更大的随机地址表空间中[2]，从而实现基于跳转表隐藏的结构图连接混淆。

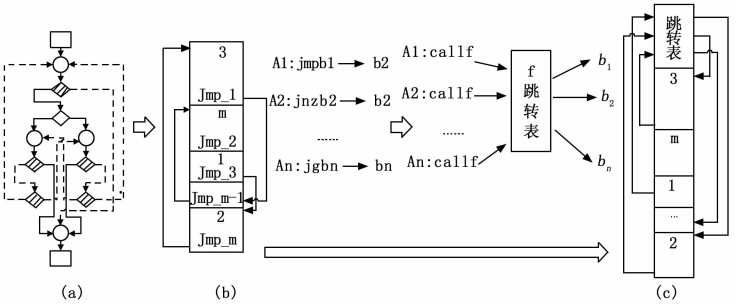

图 6　基于跳转表的结构图连接

3　实验分析

3.1　实验环境和测试用例

Windows10 家庭中文版 64 位操作系统，CPU 为 Intel（R）Core（TM）i7－8700 CPU，内存为 16 GB，主频为 3.2 GHz。

采用 IDA7.2 对 6 款测试程序进行分析，获取程序的大小和所包含的函数个数，并采用 Local Windows Debugger 调试器，对以 mspaint.exe 画图程序的图片粘贴关键代码段图 7 为例的所有测试程序进行跟踪分析，获取执行指令数目以及执行时间，基本信息如表 1 所示。

表 1 测试用例描述

源程序名称	关键代码段描述	函数数目	指令执行条数	执行时间/μs
画图程序 mspaint. exe	PasteImageClip 文件图片粘贴	3 074	1 909	0.40
记事本程序 notepad. exe	ReverseScan 逆向扫描	502	33	0.02
注册表编辑程序 regedit. exe	ImportRegFile 导入注册表文件	428	280	0.11
计算器程序 calc. exe	__wmainCRTStartup 主函数	26	350	0.12
网络探测程序 ping. exe	Wmain 主函数	43	342	0.15
网络路由程序 route. exe	Main 主函数	52	6 009	2.12

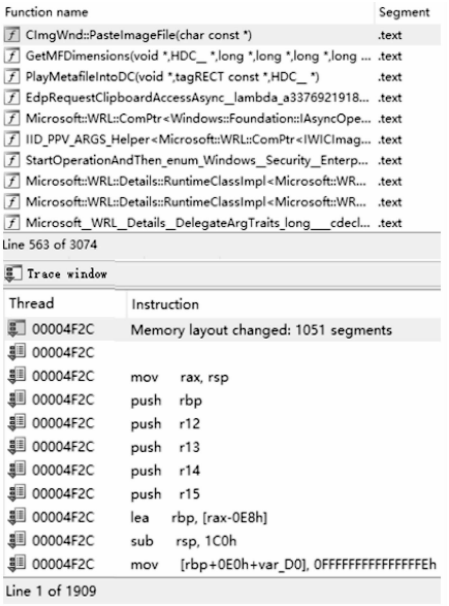

图 7 画图程序分析信息

3.2 性能分析

首先构建虚拟机代码保护原型系统 M－VM，然后再运用基于图变换的虚拟机保护方法对原型系统 M－VM 进行混淆增强变换生成 GOB－VM 虚拟机。变换过程中：设定模块切分参数为 2，数据和代码跳变中随机切分片段数目为 2，进行 3 次迭代膨胀变形，嵌套深度和宽度次数为 1 次，并采用二次映射混沌系统所构造出来的多样化不透明谓词进行模块逻辑的连接。

然后基于表 1 中的测试程序，分别采用 M－VM，GOB－VM，以及商用软件 Code Virtualizer（CV），VMProtect 对其进行保护，其中 CV 版本号为 3.0.8.0 使用的虚拟机类型为 Tiger32 White；VMP 版本号为 3.5，采用默认设置策略进行虚拟机保护。表 2 为保护前后软件大小的变化，表 3 为保护前后关键代码指令执行时间的变化。由于每次执行时间都有所不同，因此图表中执行时间为 20 次执行时间的平均值。

从表 2 和 3 可得，通过不同虚拟机框架对程序进行保护，保护后的程序无论在程序大小还是在运行时间上都会有较大的增长，如果采用虚拟机框架对程序进行保护，一般只对程序核心代码和核心数据进行保护。GOB－VM 与商用保护软件进行对比：其空间和时间的增长开销与 CV 基本持平，但远高于商用软件 VMP 和原型虚拟机系统 M－VM。在实际使用 GOB－VM 对软件进行保护的时候，如果要提高其安全强度，需要提升模块切分参数值，数据和代码跳变中随机切分片段数目，迭代膨胀变形次数，虚拟机嵌套深度和宽度次数，以及挑选安全强度更高的混沌不透明谓词，这样会使得程序的空间和时间开销有进一步的增长。

表 2 保护前后程序大小

被保护软件	保护前（kb）	保护后（kb）			
		CV3.080	VMP3.5	M－VM	GOB－VM
mspaint. exe	925	3 698	2 084	1 052	3 844
notepad. exe	177	2 446	1 397	293	2 793
regedit. exe	361	2 861	1 523	425	3 021
calc. exe	27	2 827	1 127	146	2 598
ping. exe	21	2 951	1 062	138	2 674
route. exe	24	2 725	1 084	131	2 552

表 3 保护前后关键代码执行时间

被保护软件	保护前（μs）	保护后（μs）			
		CV3.080	VMP3.5	M－VM	GOB－VM
mspaint. exe	0.40	2 547.60	477.61	153.74	3 899.25
notepad. exe	0.02	1 226.31	354.33	102.29	2 732.68
regedit. exe	0.11	2 000.11	456.22	136.55	2 823.57
calc. exe	0.12	2 057.36	498.35	132.88	3 326.22
ping. exe	0.15	2 366.47	566.51	157.87	3 567.31
route. exe	2.12	8 544.23	1 033.68	402.31	9 021.54

如果要降低虚拟机保护带给程序的时间开销，增强虚拟机保护的应用性，则需要降低模块切分参数值，数据和代码跳变中随机切分片段数目，迭代膨胀变形次数，控制虚拟机嵌套层次，以及挑选能够实现快速运算的混沌不透明谓词。

4 结束语

本文针对虚拟机核心模块连接结构的固定性，提出一种基于图变换的虚拟机安全性增强方法，方法基于虚拟机结构图变换算法，三策略组合的多样化虚拟机框架集生成算法，以及虚拟机嵌套算法，在算法实现过程中，采用混沌不透明谓词对变换后的结构进行连接，并运用跳转表对连接结构进行隐藏，基于等价变换规则的复合生成多样化代码。基于 M－VM 原型虚拟机实现了加固型 GOB－VM 虚拟机系统，通过相关分析和测试表明：基于图变换的虚拟机混淆方法，能够大大提升虚拟机系统内部结构理解的

难度，对于核心代码和数据能够有更高的保护强度，可以极大的提高数据安全性。

在实际虚拟机系统的变换实现过程中，由于涉及到多种不同的参数，如果为了更进一步提升保护的安全性，可以将模块切分更细粒度，迭代次数更多，嵌套层数更深，但随时将带来程序时间开销极大的增加，导致虚拟机保护方法应用性大大降低，但如果粗粒度切分，降低迭代次数和嵌套层数，会导致虚拟机保护方法安全性的降低。因此，下一步工作是虚拟机混淆保护方法的开销和安全性的平衡问题，并进行更大规模测试用例和参数的相关实验。

参考文献：

[1] 潘　雁，祝跃飞，林　伟. 基于指令交换的代码混淆方法 [J]. 软件学报，2019，30 (6)：1778－1792.

[2] 谢　鑫，刘粉林，芦　斌，等. Handler 混淆增强的虚拟机保护方法 [J]. 计算机工程与应用，2016，52 (15)：146－152.

[3] 谢　鑫，马　凌，陈　亮. 一种基于虚拟机 Handler 动态加解密的软件保护方法及实现 [J]. 计算机应用与软件，2017，34 (12)：321－325.

[4] 房鼎益，赵　媛，王怀军，等. 一种具有时间多样性的虚拟机软件保护方法 [J]. 软件学报，2015，26 (6)：1322－1339.

[5] 房鼎益，张　恒，汤战勇，等. 一种抗语义攻击的虚拟化软件保护方法 [J]. 工程科学与技术，2017，49 (1)：159－168.

[6] 谢　鑫，向　飞. 一种基于增强型调度器的虚拟机软件保护方法 [J]. 计算机应用与软件，2018，35 (11)：8－15.

[7] AVERBUCH A，KIPERBERG M，ZAIDENBERG N J. Truly－Protect：an efficient VM－based software protection [J]. IEEE Systems Journal，2013，7 (3)：455－466.

[8] 赫朝辉. 防篡改软件保护虚拟机的研究与实现 [D]. 西安：西北大学，2016.

[9] 吴伟民，许文锋，林志毅，等. 基于增强型虚拟机的软件保护技术 [J]. 计算机工程与科学，2014，36 (4)：655－661.

[10] 张丽娜，阎文斌. 基于虚拟机的软件保护研究与设计 [J]. 计算机工程与应用，2012，48 (26)：66－70，161.

[11] WANG H J，FANG D Y，LI G H，et al. NISLVMP：Improved virtual machine－based software protection [C] // 2013 Ninth International Conference on Computational Intelligence and Security，2013：479－483.

[12] WANG W，LI M，TANG Z Y，et al. Invalidating analysis knowledge for code virtualization protection through partition diversity [J]. IEEE Access，2019，7：169160－169173.

[13] SUK J H，LEE D H. VCF：Virtual code folding to enhance virtualization obfuscation [J]. IEEE Access，2020，8：139161－139175.

[14] SKUANG K Y，TANG Z Y，GONG X Q，et al. Enhance virtual－machine－based code obfuscation security through dynamic bytecode scheduling [J]. Computers & Security，2018，74：202－220.

[15] 侯留洋，罗森林，焦龙龙，等. 一种结合混淆思想的代码虚拟化保护方法 [J]. 科学技术与工程，2019，19 (14)：235－242.

[16] 杨　明，黄刘生. 一种采用嵌套虚拟机的软件保护方案 [J]. 小型微型计算机系统，2011，32 (2)：237－241.

[17] 谢　鑫，刘粉林，芦　斌，等. 一种基于代码并行化和虚拟机多样化的软件保护方法 [J]. 小型微型计算机系统，2015，36 (11)：2588－2593.

[18] XIE X，LIU F L，LU B. A data obfuscation based on state transition graph of mealy automata [C] // Proceedings of 10th Intelligent Computing Theory International Conference，2014：520－531.

(上接第 170 页)

[12] FEICHTENHOFER C，PINZ A，ZISSERMAN A，et al. Convolutional two－Stream network fusion for video action recognition [C] //2016 IEEE Conference on Computer Vision and Pattern Recognition. Las Vegas：IEEE，2016：1933－1941.

[13] WANG L，XIONG Y J，WANG Z，et al. Temporal segment networks：towards good practices for deep action recognition [C] // European Conference on Computer Vision (ECCV)，September，17，2016，Netherlands：2016，20－36.

[14] LIU J，SHAHROUDY A，XU D，et al. Spatio－Temporal LSTM with Trust Gates for 3D Human Action Recognition [C] // 2016 European Conference on Computer Vision (ECCV)，October 11－14，2016，Netherlands：2016，816－833.

[15] JI S，XU W，YANG M，et al. 3D convolutional neural networks for human action recognition [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence，2013，35 (1)：221－231.

[16] TRAN D，BOURDEV L，FERGUS R，et al. Learning spatiotemporal features with 3D convolutional networks [C] //2015 IEEE International Conference on Computer Vision (ICCV)，December 7－13，2015，Santiago，Chile，New York：IEEE Press，2015：4489－4497.

[17] VAROL G，LAPTEV I，SCHMID C. Long－term temporal convolutions for action recognition [J]. IEEE Transactions on Pattern Analysis & Machine Intelligence，2018，40 (6)：1510－1517.

[18] SEVILLA－lara L，LIAO Y，GUNEY F，et al. On the integration of optical flow and action recognition [C] // 2018 German Conference on Pattern Recognition (GCPR)，February 14，2018，Stuttgart：2018，281－297.

[19] HE K M，ZHANG X Y，REN S Q，et al. Deep residual learning for image recognition [C] //2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)，June 27－30，2016，Las VegasNV，USA. New York：2016，770－778.

[20] WOO S，PARK J，LEE J，et al. CBAM：convolutional block attention module [C] // European Conference on Computer Vision，2018.

[21] SOOMRO K，ZAMIR A R，SHAH M. UCF101：a dataset of 101 human actions classes from videos in the wild [J]. arXiv：1212. 0402，2012：1－7.

[22] KUEHNE H，JHUANG H，STIEFELHAGEN R，et al. HMDB51：a large video database for human motion recognition [C] // IEEE International Conference on Computer Vision. 2011：2556－2563.