本数量. $M_{ij}^{(m)}$ 则表示状态 Q_i 经过 m 步骤转移到 Q_j 的原始数据集合样本数量.

4.4 生成预测值

若未知的转移状态表示为 Q_u,则预测值的变化区间为 $Q_u=[Q_u^{\alpha},Q_u^{\beta}]$,预测值 $\hat{Y}(k)$ 取该区间的中点,则

$$\hat{Y}(k)=\frac{Q_u^{\alpha}+Q_u^{\beta}}{2}.$$

4.5 检验预测精度

预测精度主要通过后验差比值及小误差频率来完成,后验差比值生成 C 值越小越好,而小误差频率 P 值越大越好,需考虑 C<0.35 且 P>0.95.

4.6 拟合预测方向

以上预测过程的主要目的是通过该模型获取目标虚拟化平台中模糊测试数据的生成方向及内容,考虑以下几种情况作为预测方向的拟合特征.

1. $\hat{Y}(k)_i \rightarrow IaaS\_DDoS\_Vul_i, i\in\{TP,TD,TF,TB\}$;
   //存在预测方向拟合虚拟化拒绝服务规则
2. $\hat{Y}(k)_i \rightarrow IaaS\_Escape\_Vul_i, i\in\{TP,TD,TF,TB\}$;
   //存在预测方向拟合虚拟化逃逸规则
3. $\hat{Y}(k)\rightarrow IaaS\_Escape\_Vul$ and $\hat{Y}(k)\rightarrow IaaS\_DDoS\_Vul, i\in\{TP,TD,TF,TB\}$.
   //存在预测方向同时拟合虚拟化拒绝服务规则与逃逸规则

4.7 指导模糊测试

本文模糊测试所采用的基本方法是面向虚拟化平台中 3 种典型的虚拟化过程展开,包括 CPU 虚拟化、内存虚拟化及 IO 虚拟化.首先,基于源码静态审计的方法确定各虚拟化功能模块的函数调用关系、函数调用接口、函数参数约束条件等;而后,通过动态调试分析虚拟化过程中的动态函数调用路径及函数执行结果,并通过与静态源码的比对确定函数相关信息;而后,依次遍历确定后的目标测试函数,并将已收集的函数入口参数取值范围及多重函数调用过程作为模糊测试对象,基于标准化的随机化种子生成方法在参数取值范围内及多重函数调用路径间生成模糊测试用例;而后,大量冲击目标系统,观察系统反馈结果.以上基于灰度马尔可夫链的预测模型将生成系统状态预测值,通过预测方向的拟合结果,可判决当前虚拟化模块下的模糊测试方向,从而决定是否在拟合的漏洞发现预测方向下继续执行模糊测试,或自适应调整当前模糊测试输入参数及函数调用关系,使得计算后的预测方向与正确的拟合方向相同.针对不同虚拟化模块中的不同虚拟化功能,在决定自适应调整后需改变不同的参数及不同的函数调用关系,因此,该过程需面向虚拟化实现机制加入部分人工干预.

5 原型系统实现

本文面向 IaaS 层的几种典型虚拟化平台部署并实现原型系统 VirtualFuzz,主要基于 GuestOS 与 HostOS 的通信模型进行源码审计、动态调试与污点分析,经过源代码的静态审计获取静态函数调用关系与静态关联数据结构,在动态调试过程中,补充过程间调用及动态填充数据结构,基于污点分析修正动态函数调用关系及输入数据约束条件在未知输入数据约束条件下的异常情况,最终按模块、直接调用函数、间接调用函数、过程间调用函数的顺序依次完成模糊测试过程.其中用到的源码审计数据来源主要是各实验版本中开源项目(vmware, bochs 为非开源项目,因此该两类实验对象中静态分析主要通过逆向分析完成),动态调试主要使用 windows 平台及 linux 平台下典型的调试工具:windbg(版本号:6.12.2.633),GDB(版本号:7.3)等,污点跟踪则主要通过静态分析工具插件开发(IDA6.8 版本)、调试器插件编写(windbg 插件)、自研的内存查看及标记工具完成.核心功能中的模糊测试部分则是根据以上自适应指导过程具体在各 IaaS 层虚拟化产品中的网络数据包接口、外部命令接口、CPU 指令接口中展开模糊测试,挖掘虚拟化过程在接受外部网络数据包、HostOS 与 GuestOS 通信命令、

CPU 指令集实现过程中的拒绝服务与逃逸漏洞.同时,基于灰度马尔可夫为模糊测试过程生成预测值,实时修正模糊测试的输入及方向,避免错误或低效的测试开销.具体过程如图 2 所示.

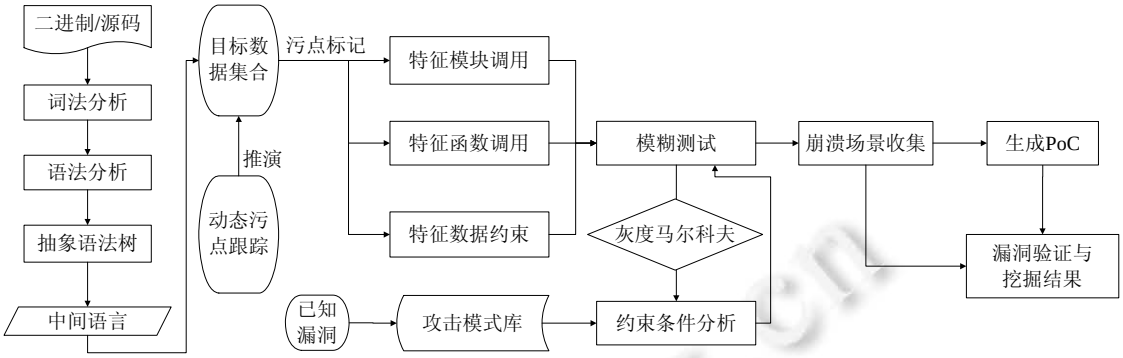


Fig.2 Design flow of VirtualFuzz

图 2 VirtualFuzz 设计流程

由图 2 可知:系统首先对实验对象的开源代码或二进制文件展开词法分析及语法分析,并通过抽象语法树的方法获得树形表达结果,为转化为中间语言表达提供支持.而后,基于 Valgrind 中的 VEX-IR 中间语言表达实现目标数据集合的污点标记,并与 Windows 及 Linux 平台下的动态调试器插件进行通信,完成动态污点跟踪过程.随后,针对 3 种特征调用展开模糊测试,对崩溃场景中寄存器取值进行收集,最终生成漏洞利用程序(proof of concept,简称 PoC).

6 实 验

实验主要针对 5 类应用范围最广泛的 IaaS 层中虚拟化平台及工具展开测试,包括 Xen,KVM,Qemu,Bochs,Vmware,将跨平台覆盖虚拟化实现过程中大量关键技术.首先,基于静态与动态分析过程建立目标测试数据集,并在此基础上拓展建立面向拒绝服务与逃逸的攻击模式库;而后部署所设计的模糊测试系统,基于攻击模式库依次遍历目标测试数据集中的关键控制流与数据结构,并实时生成灰度马尔可夫预测值,比对判决规则后指导模糊测试方向,达到自适应效果.具体的实验环境参数如下所示.

1) 硬件平台
   - 处理器:Intel(R)Core2Duo CPU E7500@2.13GHz;
   - 内存:32.00GB;
   - 显卡:NVIDIA Ge Force 9600 GSO 512.
2) 软件平台
   - 操作系统:Windows 7 旗舰版 Service Pack1;SUSE Linux Enterprise Server 12;
   - 开发环境:VS 2013;GCC 7.1.
3) 测试对象
   - Xen:3.0.1--4.5.x(基于 Linux 平台);
   - KVM:11-17(基于 Linux 平台);
   - Qemu:2.2.1--2.7.x(基于 Linux 平台);
   - Vmware:10.0.4--12.0.x(基于 Windows 平台);
   - Bochs:2.5.0--2.6.x(基于 Windows 平台).

6.1 目标测试数据集

实验基于目标平台的静态与动态分析进行类型推演,收集关键函数、数据结构及相互间的交叉关系,尤其

针对已曝光漏洞的功能模块及相关函数,需基于漏洞分析生成原始的目标测试结合,类型推演后去除已曝光的数据约束关系,将相关信息添加到目标测试数据集中.此处以漏洞 CVE-2015-3456 为例,展示如何从静态与动态分析及漏洞场景分析中生成目标测试集合,并完成类型推演,具体过程如图 3 所示.

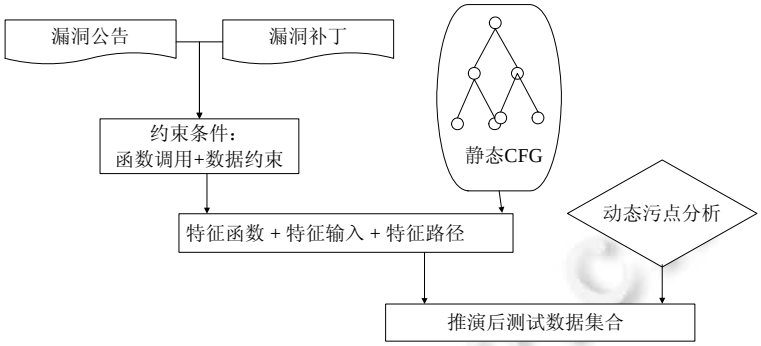


Fig.3 Generation of target test sets

图 3 目标测试集合生成过程

6.2 自适应模糊测试过程分析

工具针对第 6.1 节中推演所得的目标测试集合,展开自适应的模糊测试过程.面向第 6.1 节中推演所得的目标测试集合,逐个平台、逐个模块、逐个函数及逐个约束入口进行模糊测试,同时计算基于灰度马尔可夫的预测值,以拟合拒绝服务及逃逸漏洞判决规则为指导,调整模糊测试方向.鉴于目标测试数据集中元素数量庞大,本文对各平台下的不同虚拟化模块进行测试方向与预测过程的统计分析,具体统计数据如图 4 所示.

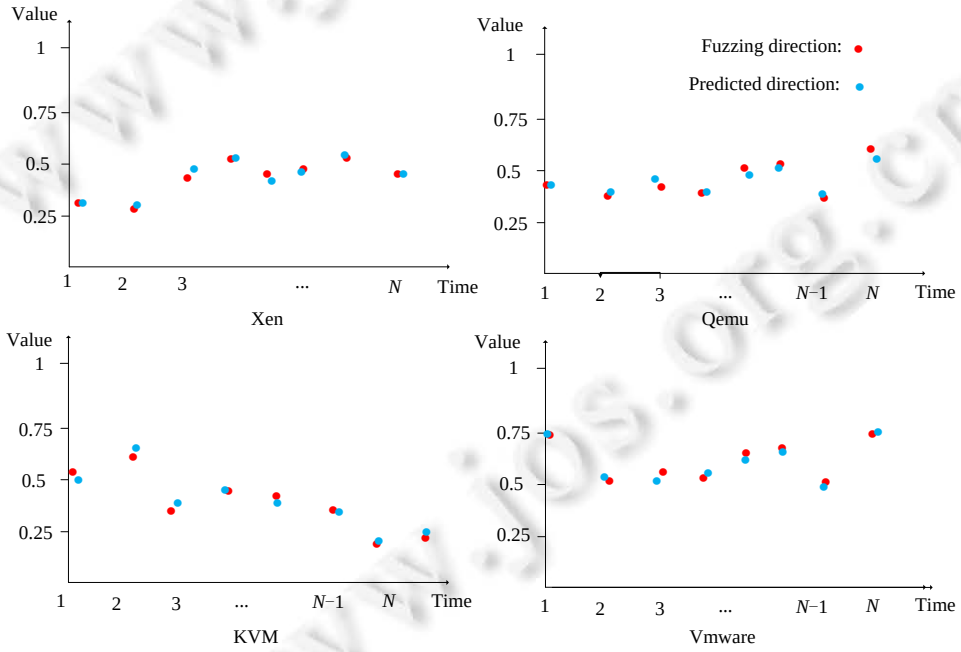


Fig.4 Prediction statistic for adaptive fuzzing test

图 4 自适应模糊测试预测统计

由图 4 可知:实验对 4 类平台中的自适应模糊测试过程进行数据统计,将目标系统中共 486 538 个采样点进行分类,并选择其中有限的 100 个采样点进行趋势绘图,其取值来自于 $X=\{x.TB,x.TP,x.TF,x.TD,x.CEm,x.CP,$

x.CEx,x.EP}集合在多属性决策方法中的理想优基点法下的取值,取值从 0 到 1,根据时间点 *N* 下几个具备典型漏洞特征的采样点作为记录,可绘制如图 4 所示的模糊测试方向取值点及灰度马尔可夫预测取值点.具体来说,以 Xen 虚拟化平台为例,漏洞特征规则下的模糊测试采样点大都分布在 0.25~0.6 之间,而马尔可夫预测点与其误差在[0.13,0.22]内,针对第 3 采样点处出现最大误差,该点在预测模型下显示为无漏洞特征,而模糊测试过程中判决为疑似拒绝服务漏洞.后经人工排查,该点处的拒绝服务漏洞不具备可用的数据约束路径.同理可分析 Qemu 平台下的对比结果,在所展示的第 7 采样点处,多属性聚合值为最小,取值为 0.572,表征在理想优基点模型中,该点的漏洞特征与其他疑似漏洞相比最小,即最不可能判决为漏洞.但该点处模糊测试方向与预测方向拟合效果较好,误差仅为 0.08,且在预测模型中被判决为疑似拒绝服务漏洞.后经人工排查可知,该点处无法触发拒绝服务漏洞利用效果,证明实际的模糊测试所生成的数据约束路径无效.同理对比分析 KVM 和 Vmware 平台,KVM 整体安全性分析结果较差,多处存在差拟合效果的漏洞评估.Vmware 平台的拟合效果较好,误差控制在 0.03~0.14 之间,可被认为具备较高的安全级别,且模糊测试中判决得到的漏洞个数较少,经人工确认后的可用漏洞更少.同理可展示 Bochs 虚拟化环境下的数据分析结果,其安全性与 Vmware 基本同级,限于篇幅,不再展开讲解.

6.3 漏洞模式判决

对目标系统中通过自适应模糊测试获得的漏洞进行统计并整理,鉴于 Vmware 及 Bochs 平台下并未发现有效漏洞,因此主要统计前 3 类平台(Xen,KVM,Qemu)中的漏洞数量,主要包括两类:(1) 系统认定的漏洞;(2) 经人工排查后确定可利用的漏洞.第 1 类漏洞基于模糊测试中的随机化输入数据及系统崩溃场景记录判决;第 2 类需在第 1 类的基础上回溯分析触发场景的数据约束入口,保证在固定的数据约束条件下可生成拒绝服务或逃逸场景.具体数据见表 1,对 3 类平台进行总结,分别获得拒绝服务漏洞场景 5,4,7 个,分别获得逃逸场景 2,2,4 个,分别覆盖 Xen,KVM,Qemu 平台.

Table 1 Vulnerability judgement results

表 1 漏洞判决结果

种类	平台		
	Xen	KVM	Qemu
拒绝服务	31(5)	22(4)	54(7)
逃逸	20(2)	25(2)	37(4)

6.4 已知漏洞验证及未知漏洞发现

针对已获得的漏洞个数及类型,本文总结已发现的漏洞见表 2,共计 24 个,包括 Xen 平台中的 7 个、KVM 平台中的 6 个以及 Qemu 平台中的 11 个,依次命名为 Vul_A 到 Vul_X.由于原型系统 VirtualFuzz 中使用的静态分析结果和攻击模式库均是从开放源码和已知漏洞中推演而来,因此,该 24 个漏洞中包含用于建模的若干个已知 CVE 漏洞,同时也发现了大量未知的高风险漏洞.如表 2 所示,系统所挖掘漏洞分别对应到 3 个主流虚拟化平台,已针对 Xen 发现 2 个未知漏洞,分别对应到 IO 模块的网卡处理函数及声卡处理函数中,目前已申请 CVE 号,处于待审批状态;KVM 虚拟化环境中主要完成了已知漏洞的验证过程,包括 4 个拒绝服务漏洞及 2 个逃逸漏洞;Qemu 虚拟化中获得了 4 个未知漏洞,其中有 3 个已拿到 CVE 号,另一个是本文工作在最近所发现,因此也在申请 CVE 中,具体模块位于 IO 处理过程的网卡数据收包函数中,其余主要验证了系统中部分的已知漏洞.另外,从描述中可知:还有部分漏洞属于已发布但未分配 CVE 号的情况,此类漏洞可通过漏洞描述基本确定所得漏洞是否与其特征相同.

以 CVE-2016-2841 为例,展开漏洞挖掘的完整过程如下:(1) 经测试目标推演后,Qemu 源码中测试数据集中网络接口集合主要来自于 hw/net 目录下的各类源码对网卡的模拟,将其输入原型系统,通过词法分析、语法分析、抽象语法树处理后转化为 VEX-IR 中间语言进行表达,基于 GDB 动态调试过程在中间语言表达中插入动态监控点,重点标记特征模块、特征函数及特征数据约束条件;(2) 而后,基于已知 ne2000 网卡中漏洞的特征点完成相关定义及规则的赋值与判决;(3) 在建立灰系统后,使用马尔可夫链拟合网卡在接受数据包过程中可能诱发拒绝服务及逃逸的数据约束条件;(4) 在面向特征模块、函数及数据条件的随机化测试过程中加入马尔可

夫拟合过程的指导作用,使得随机化过程按照灰色马尔可夫拟合方向进行数据变异及数据生成;(5) 最终,通过构造特征化的 PSTARTPSTOP 寄存器使目标 VM 拒绝服务.

Table 2 Verification of known vulnerabilities and discovery of unknown vulnerabilities

表 2 验证已知漏洞及发现未知漏洞

漏洞名	平台	类型	模块	已知/未知
Vul_A	Xen	拒绝服务	CPU	已知:CVE-2015-7812
Vul_B	Xen	拒绝服务	IO	已知:CVE-2015-0268
Vul_C	Xen	拒绝服务	IO	已知:CVE-2014-2580
Vul_D	Xen	拒绝服务	IO	已知:CVE-2014-2580
Vul_E	Xen	拒绝服务	IO	**未知,已申请 CVE**
Vul_F	Xen	逃逸	Memory	已知:CVE-2015-7835
Vul_G	Xen	逃逸	IO	**未知,已申请 CVE**
Vul_H	KVM	拒绝服务	IO	已知:CVE-2015-6815
Vul_I	KVM	拒绝服务	IO	已知:CVE-2014-7842
Vul_J	KVM	拒绝服务	Memory	已知:CVE-2016-4440
Vul_K	KVM	拒绝服务	IO	已知:无 CVE 号
Vul_L	KVM	逃逸	IO	已知:CVE-2015-6815
Vul_M	KVM	逃逸	IO	已知:CVE-2015-0239
Vul_N	Qemu	拒绝服务	CPU	已知:CVE-2016-2858
Vul_O	Qemu	拒绝服务	Memory	已知:无 CVE 号
Vul_P	Qemu	拒绝服务	IO	**未知:获得 CVE-2016-2841**
Vul_Q	Qemu	拒绝服务	IO	**未知:获得 CVE-2016-0749**
Vul_R	Qemu	拒绝服务	IO	**未知:获得 CVE-2016-2150**
Vul_S	Qemu	拒绝服务	IO	已知:CVE-2014-3672
Vul_T	Qemu	拒绝服务	IO	已知:CVE-2016-2391
Vul_U	Qemu	逃逸	IO	已知:CVE-2016-2391
Vul_V	Qemu	逃逸	IO	已知:CVE-2015-7504
Vul_W	Qemu	逃逸	IO	**未知,已申请 CVE**
Vul_X	Qemu	逃逸	IO	已知:CVE-2015-3456

6.5 性能分析与比较

目前,工业界及学术界均有较多的模糊测试工具,为了衡量 VirtualFuzz 原型系统的性能开销,本文对比了 3 种典型的商业化模糊测试工具,主要计算目标系统在模糊测试运行前后的性能损耗.如表 3 所示,AFL 主要面向应用程序展开智能化模糊测试,可有效地获得测试中错误位置并绕行,性能损耗约为 46%;Trinity 是面向 Android 环境的模糊测试工具,可同时兼容 Android 应用层及内核层,性能损耗约为 36%;Peach 是近年来安全研究人员及漏洞挖掘人员常用的模糊测试工具,适用范围较广,功能强大且界面友好,其性能损耗约为 41%;本文中的原型系统 VirtualFuzz 面向 5 种虚拟化平台时的平均损耗达到 23.69%,极大地优化了已有工具的性能开销,可知基于灰度的马尔可夫自适应过程为系统性能的提升做出了巨大的贡献,后续还有一定的优化空间.

Table 3 Performance comparison between prototype and other fuzzing tools

表 3 原型系统与其他模糊测试工具的性能比较

污点跟踪方案	测试前(μs)	测试中(μs)	平均性能损耗(%)
AFL	47.229 1	88.230 6	46.12
Trinity	66.441 7	93.219 1	36.61
Peach	53.693 1	79.337 8	41.35
VirtualFuzz	59.844 9	81.321 5	23.69

7 结 论

本文面向 IaaS 层中的虚拟化平台设计并实现了一种漏洞挖掘方法,基于静态源码审计及动态污点跟踪实现定向模块及函数的模糊测试,并设计了一种面向模糊测试的自适应监督方法:灰度马尔可夫预测模型,在预测值的引导下实时调整模糊测试的方向和内容,实现有效且快速的虚拟化漏洞验证与挖掘,最终针对多种虚拟化环境实现了已知漏洞的验证及未知漏洞的挖掘,在一定程度上证实了该部分工作的有效性.后续需要深入拓展

各模块及函数的调用关系,并进一步优化模糊测试的效率,希望能够在不依托已知漏洞的情况下实现更大范围的自适应模糊测试方法.

References:

[1] Wang WX, Zhang J, Chang Q, Gu ZJ. Research on the security problem of cloud computing virtualization platform. Netinfo Security, 2016,2016(9):163−168 (in Chinese with English abstract).

[2] Gong Y, Li C, Wu W. Research on the security technology in virtualization. Netinfo Security, 2016,2016(9):73−78 (in Chinese with English abstract).

[3] Ma W, Han Z, Cheng Y. Research on multi-level management mechanism in trusted cloud computing. Netinfo Security, 2015, 2015(7):20−25 (in Chinese with English abstract).

[4] Shan GD, Dai YX, Wang H. Study on computer vulnerability taxonomy. Computer Engineering, 2002,28(10):3−6 (in Chinese with English abstract). [doi: 10.3969/j.issn.1000-3428.2002.10.002]

[5] Chen YC, Chen GQ, Chen ZM, Wan N. A binary code analysis of vulnerability scanning method for SDN smart grid. Netinfo Security, 2016,2016(7):35−39 (in Chinese with English abstract).

[6] Wu SZ, Guo T, Dong GW, Wang JJ. Software vulnerability analyses: A road map. Journal of Tsinghua University (Science and Technology), 2012,52(10):1309−1319 (in Chinese with English abstract).

[7] Li ZJ, Zhang JX, Liao XK, Ma JX. Survey of software vulnerability detection techniques. Chinese Journal of Computers, 2015, 38(4):717−732 (in Chinese with English abstract).

[8] Guo X, Wang P. Technique of cooperative reverse reasoning in related path static analysis. Ruan Jian Xue Bao/Journal of Software, 2015,26(1):1−13 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4658.htm [doi: 10.13328/j.cnki.jos.004658]

[9] Gan ST, Qin XJ, Chen ZN, Wang LZ. Software vulnerability code clone detection method based on characteristic metrics. Ruan Jian Xue Bao/Journal of Software, 2015,26(2):348−363 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4786.htm [doi: 10.13328/j.cnki.jos.004786]

[10] Ganapathy V, Jha S, Chandler D, Melski D, Vitek D. Buffer overrun detection using linear programming and static analysis. In: Proc. of the ACM Conf. on Computer and Communications Security. New York: Academic Press, 2003. 345−354. [doi: 10.1145/948109.948155]

[11] Zhang DH, Liu DG, Wang WH, Lei J, Kung D, Csallner C. Testing C programs for vulnerability using trace-based symbolic execution and satisfiability analysis. In: Proc. of the Int'l Conf. on Dependable Systems and Networks (DSN 2010). New York: IEEE Computer Society, 2010. 321−338.

[12] Wang WG, Zeng QK, Sun H. Dynamic symbolic execution method oriented to critical operation. Ruan Jian Xue Bao/Journal of Software, 2016,27(5):1230−1245 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5027.htm [doi: 10.13328/j.cnki.jos.005027]

[13] Cui ZQ, Wang LZ, Li XD. Target-Directed concolic testing. Chinese Journal of Computers, 2011,34(6):953−964 (in Chinese with English abstract).

[14] Li X, Zhou Y, Li MC, Chen YJ, Xu GQ, Wang LZ, Li XD. Automatically validating static memory leak warnings for C/C++ programs. Ruan Jian Xue Bao/Journal of Software, 2017,28(4):827−844 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5189.htm [doi: 10.13328/j.cnki.jos.005189]

[15] Sun H, Li HP, Zeng QK. Statically detect and run-time check integer-based vulnerabilities with information flow. Ruan Jian Xue Bao/Journal of Software, 2013,24(12):2767−2781 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4385.htm [doi: 10.3724/SP.J.1001.2013.04385]

[16] Zou Q, Wu M, Hu WW, Zhang LB. An instrument-analysis framework for adaptive prefetch optimization in JVM. Ruan Jian Xue Bao/Journal of Software, 2008,19(7):1581−1589 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/19/1581.htm [doi: 10.3724/SP.J.1001.2008.01581]

[17] Ma JX, Li ZJ, Zhang T, Shen D, Zhang ZK. Taint analysis method based on offline indices of instruction trace. Ruan Jian Xue Bao/Journal of Software, 2017,28(9):2388−2401 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5179.htm [doi: 10.13328/j.cnki.jos.005179]

[18] Yang Z, Yin LH, Duan MY, Wu JY, Jin SY, Guo L. Generalized taint propagation model for access control in operation systems. Ruan Jian Xue Bao/Journal of Software, 2012,23(6):1602−1619 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4083.htm [doi: 10.3724/SP.J.1001.2012.04083]

[19] Sang KC, Woo M, Brumley D. Program-Adaptive mutational fuzzing. In: Proc. of the Security and Privacy. New York: IEEE Computer Society, 2015. 725−741. [doi: 10.1109/SP.2015.50]

[20] Ma JX, Zhang T, Li ZJ, Zhang JX. Improved fuzzy analysis methods. Journal of Tsinghua University (Science and Technology), 2016,56(5):478−483 (in Chinese with English abstract).

[21] Godefroid P, Kiezun A, Levin MY. Grammar-Based whitebox fuzzing. In: Proc. of the ACM Sigplan Conf. on Programming Language Design and Implementation. New York: Academic Press, 2008. 206−215. [doi: 10.1145/1375581.1375607]

[22] Li WM, Yu JQ, Ai SB. PyFuzzer: Automatic in-memory fuzz testing method. Journal on Commnications, 2013,34(Z2):64−68 (in Chinese with English abstract).

[23] Ouyang YJ, Wei Q, Wang QX, Yin ZX. Intelligent fuzzing based on exception distribution steering. Journal of Electronics & Information Technology, 2015,37(1):143−149 (in Chinese with English abstract).

[24] Huang L, Feng DG, Lian YF, Chen K, Zhang YJ, Liu YL. Method of DDoS countermeasure selection based on multi-attribute decision making. Ruan Jian Xue Bao/Journal of Software, 2015,26(7):1742−1756 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4673.htm [doi: 10.13328/j.cnki.jos.004673]

[25] Wang L, Yang XJ, Wang J, Luo Y. Automatedly checking function execution context of kernel programs in operation systems. Ruan Jian Xue Bao/Journal of Software, 2007,18(4):1056−1067 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/18/1056.htm [doi: 10.1360/jos181056]

[26] Feng DG, Zhang M, Zhang Y, Xu Z. Study on cloud computing security. Ruan Jian Xue Bao/Journal of Software, 2011,22(1): 71−83 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3958.htm [doi: 10.3724/SP.J.1001.2011.03958]

[27] Lin C, Su WB, Meng K, Liu Q, Liu WD. Cloud computing security: Architecture, mechanism and modeling. Chinese Journal of Computers, 2013,36(9):1765−1784 (in Chinese with English abstract).

[28] Zhang YQ, Wang XF, Liu XF, Liu L. Survey on cloud computing security. Ruan Jian Xue Bao/Journal of Software, 2016,27(6): 1328−1348 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5004.htm [doi: 10.13328/j.cnki.jos.005004]

[29] Zhu M, Tu BB, Meng D. The security research of virtualization software stack. Chinese Journal of Computers, 2017,40(2):481−504 (in Chinese with English abstract). [doi: 10.11897/SP.J.1016.2017.00481]

[30] Elhage NV. A KVM Guest→Host privilege, escalation exploits. In: Proc. of the BlackHat USA. New York: IEEE Computer Society, 2011. 1−12.

[31] Wang G, Estrada ZJ, Pham C, Kalbarczyk Z. Hypervisor introspection: A technique for evading passive virtual machine monitoring. In: Proc. of the 2016 Usenix Security Symp. Boca Raton: CRC Press, 2016. 207−225.

[32] Panuccio A, Bicego M, Murino V. A Hidden Markov Model-Based Approach to Sequential Data Clustering Structural, Syntactic, and Statistical Pattern Recognition. Vol.1, Berlin, Heidelberg: Springer-Verlag, 2002. 734−743. [doi: 10.1007/3-540-70659-3_77]

[33] Rabiner LR. A tutorial on hidden Markov models and selected applications on speech recognition. Readings in Speech Recognition, 1990,77(2):267−296.

附中文参考文献:

[1] 王文旭,张健,常青,顾兆军.云计算虚拟化平台安全问题研究.信息网络安全,2016,2016(9):163−168.

[2] 宫月,李超,吴薇.虚拟化安全技术研究.信息网络安全,2016,2016(9):73−78.

[3] 马威,韩臻,成阳.可信云计算中的多级管理机制研究.信息网络安全,2015,2015(7):20−25.

[4] 单国栋,戴英侠,王航.计算机漏洞分类研究.计算机工程,2002,28(10):3−6. [doi: 10.3969/j.issn.1000-3428.2002.10.002]

[5] 陈颖聪,陈广清,陈智明,万能.面向智能电网 SDN 的二进制代码分析漏洞扫描方法研究.信息网络安全,2016,2016(7):35−39.

[6] 吴世忠,郭涛,董国伟,王嘉捷.软件漏洞分析技术进展.清华大学学报自然科学版,2012,52(10):1309−1319.

[7] 李舟军,张俊贤,廖湘科,马金鑫.软件安全漏洞检测技术.计算机学报,2015,38(4):717−732.

[8] 郭曦,王盼.相关路径静态分析中协同式逆向推理方法.软件学报,2015,26(1):1−13. http://www.jos.org.cn/1000-9825/4658.htm [doi: 10.13328/j.cnki.jos.004658]

[9] 甘水滔,秦晓军,陈左宁,王林章.一种基于特征矩阵的软件脆弱性代码克隆检测方法.软件学报,2015,26(2):348−363. http://www.jos.org.cn/1000-9825/4786.htm [doi: 10.13328/j.cnki.jos.004786]

[12] 王伟光,曾庆凯,孙浩.面向危险操作的动态符号执行方法.软件学报,2016,27(5):1230−1245. http://www.jos.org.cn/1000-9825/5027.htm [doi: 10.13328/j.cnki.jos.005027]

[13] 崔展齐,王林章,李宣东.一种目标制导的混合执行测试方法.计算机学报,2011,34(6):953−964. [doi: 10.3724/SP.J.1016.2011.00953]

[14] 李筱,周严,李孟宸,陈园军,Xu GQ,王林章,李宣东.C/C++程序静态内存泄漏警报自动确认方法.软件学报,2017,28(4):827−844. http://www.jos.org.cn/1000-9825/5189.htm [doi: 10.13328/j.cnki.jos.005189]

[15] 孙浩,李会朋,曾庆凯.基于信息流的整数漏洞插装和验证.软件学报,2013,24(12):2767−2781. http://www.jos.org.cn/1000-9825/4385.htm [doi: 10.3724/SP.J.1001.2013.04385]

[16] 邹琼,伍鸣,胡伟武,章隆兵.基于插桩分析的 Java 虚拟机自适应预取优化框架.软件学报,2008,19(7):1581−1589. http://www.jos.org.cn/1000-9825/19/1581.htm [doi: 10.3724/SP.J.1001.2008.01581]

[17] 马金鑫,李舟军,张涛,沈东,章张锴.基于执行踪迹离线索引的污点分析方法.软件学报,2017,28(9):2388−2401. http://www.jos.org.cn/1000-9825/5179.htm [doi: 10.13328/j.cnki.jos.005179]

[18] 杨智,殷丽华,段冰毅,吴金宇,金舒原,郭莉.基于广义污点传播模型的操作系统访问控制.软件学报,2012,23(6):1602−1619. http://www.jos.org.cn/1000-9825/4083.htm [doi: 10.3724/SP.J.1001.2012.04083]

[20] 马金鑫,张涛,李舟军,张江霄.Fuzzing 过程中的若干优化方法.清华大学学报(自然科学版),2016,56(5):478−483.

[22] 李伟明,于俊清,艾少波.PyFuzzer:自动化高效内存模糊测试方法.通信学报,2013,34(Z2):64−68.

[23] 欧阳永基,魏强,王清贤,尹中旭.基于异常分布导向的智能 Fuzzing 方法.电子与信息学报,2015,37(1):143−149.

[24] 黄亮,冯登国,连一峰,陈恺,张颖君,刘玉岭.一种基于多属性决策的 DDoS 防护措施遴选方法.软件学报,2015,26(7):1742−1756. http://www.jos.org.cn/1000-9825/4673.htm [doi: 10.13328/j.cnki.jos.004673]

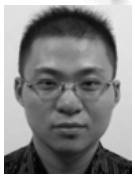
[25] 汪黎,杨学军,王戟,罗宇.操作系统内核程序函数执行上下文的自动检验.软件学报,2007,18(4):1056−1067. http://www.jos.org.cn/1000-9825/18/1056.htm [doi: 10.1360/jos181056]

[26] 冯登国,张敏,张妍,徐震.云计算安全研究.软件学报,2011,22(1):71−83. http://www.jos.org.cn/1000-9825/3958.htm [doi: 10.3724/SP.J.1001.2011.03958]

[27] 林闯,苏文博,孟坤,刘渠,刘卫东.云计算安全:架构、机制与模型评价.计算机学报,2013,36(9):1765−1784.

[28] 张玉清,王晓菲,刘雪峰,刘玲.云计算环境安全综述.软件学报,2016,27(6):1328−1348. http://www.jos.org.cn/1000-9825/5004.htm [doi: 10.13328/j.cnki.jos.005004]

[29] 朱民,涂碧波,孟丹.虚拟化软件栈安全研究.计算机学报,2017,40(2):481−504. [doi: 10.11897/SP.J.1016.2017.00481]



沙乐天(1985－),男,江苏南京人,博士,讲师,CCF 专业会员,主要研究领域为软件安全,漏洞挖掘,物联网攻防.

肖甫(1980－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为物联网,传感网.

杨红柯(1983－),男,工程师,主要研究领域为网络安全,漏洞挖掘.

喻辉(1972－),男,工程师,主要研究领域为网络安全,漏洞挖掘.

王汝传(1943－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为物联网.