面向进程控制流劫持攻击的拟态防御方法

潘传幸[1]，张铮[1]，马博林[2]，姚远[1]，季新生[2]
（1. 数学工程与先进计算国家重点实验室，河南 郑州 450001；2. 国家数字交换系统工程技术研究中心，河南 郑州 450002）

摘 要：为了防御进程控制流劫持攻击，从漏洞利用的角度对攻击过程建立了威胁模型，提出了截断关键漏洞利用环节的“要塞”防御。在研究拟态防御原理的基础上提出了进程的拟态执行模型，并对该模型进行了分析与有效性证明，拟态执行能够有效截断控制流劫持的攻击实施过程；实现了拟态执行的原型系统 MimicBox，并对 MimicBox 进行了有效性验证实验、性能测试和对比评估。有效性验证实验表明，MimicBox 可以有效防御绝大部分基于已知类型二进制漏洞的控制流劫持攻击；性能评估结果表明，MimicBox 对 CPU 密集型程序带来的额外性能开销不会超过 13%；对比评估结果表明，拟态执行相对于控制流完整性防御来说，是一种较有效实用的主动防御方案。

关键词：控制流劫持；拟态防御；拟态执行；原型系统；评估测试
中图分类号：TP393.0
文献标识码：A
DOI: 10.11959/j.issn.1000−436x.2021013

Method against process control-flow hijacking based on mimic defense

PAN Chuanxing[1], ZHANG Zheng[1], MA Bolin[2], YAO Yuan[1], JI Xinsheng[2]
1. State Key Laboratory of Mathmatical Engineering and Advanced Computing, Zhengzhou 450001, China
2. National Digital Switching System Engineering & Technological Research Center, Zhengzhou 450002, China

Abstract: To defeat the attack of process control flow hijacking, a threat model was established from the point of vulnerability utilization, and the fortress defense to cut off the key vulnerability utilization path was proposed. On the basis of studying the principle of mimic defense, a threat model of process mimic execution was proposed, and the threat model was analyzed and proved to be effective. Mimic execution could effectively cut off the attack path of control flow hijacking. The ptototype of mimic execution, MimicBox, was implemented. The validation experiment shows that MimicBox can effectively defend against most control flow hijacking attacks based on known binary vulnerabilities. The performance evaluation result shows that the overhead MimicBox lead to is less than 13% on CPU-intensive programs. The Comparative evaluation result shows that mimic execution is a more effective and practical active defense method compared with control flow integrity.

Keywords: control-flow hijacking, mimic defense, mimic execution, prototype, evaluation

1 引言

进程控制流劫持是一种常见的攻击手段，其通常利用缓冲区溢出[1]等二进制漏洞篡改进程的控制流，从而进行程序正常功能之外的恶意操作[2]。控制流劫持攻击的危害非常大，控制流劫持的目的往往是获取目标机器的控制权，然后提取系统特权对目标机器实现全面控制。根据 CWE

收稿日期：2020−08−26；**修回日期**：2020−11−05
通信作者：张铮，ponyzhang@126.com
基金项目：国家自然科学基金资助项目（No.61521003）；国家重点研发计划基金资助项目（No.2018YFB0804003）
Foundation Items: The National Natural Science Foundation of China (No.61521003), The National Key Research and Development Program of China (No.2018YFB0804003)

Top25[3]的排名，内存缓冲区溢出仍然是最危险的软件错误。

控制流劫持攻击可以分为代码注入和代码重用两大类[2]。代码注入类攻击利用程序本身的输入功能向进程的虚拟地址空间注入恶意代码，然后通过缓冲区溢出等方式覆盖返回地址，实现控制流劫持。代码重用类攻击不需要注入代码，而是利用程序自身的代码片段完成攻击。为了防御进程控制流劫持，研究人员和工程技术人员提出了许多防御手段。早期，控制流劫持攻击通过代码注入实现，为了防御代码注入攻击，研究人员提出了数据执行保护（DEP, data execution prevention）、栈不可执行[4]、栈监控[5]、地址空间布局随机化（ASLR, address space layout randomization）[6]等方法。然而，这些方法却无法防御面向返回编程（ROP, return oriented programming）[7]等基于代码重用的控制流劫持[8-11]。控制流完整性（CFI, control-flow integrity）[12-13]是防御代码重用较有效的方法，其通过对二进制可执行程序进行静态分析，建立可信的控制流图，在执行时限制控制流转移。但是，对于信息泄露和面向数据编程（DOP, data oriented programming）[14]攻击，即使细粒度的 CFI 也难以起到防御作用。

拟态防御是我国科研团队提出的针对未知漏洞后门的主动防御思想[15-16]，核心是动态异构冗余（DHR, dynamic heterogeneous redundancy）构造[17]，如图 1 所示。各个执行体之间功能等价，但受到攻击后的行为表现各异，表决器通过对比在线执行体集中各执行体之间的行为是否一致来判断是否遭到攻击。一旦表决器判定为遭到攻击，则调用动态选择算法从离线的异构构件集合中挑选合适的构件对在线执行体进行替换，这个过程称为拟态变换。输入代理与表决器中间的部分称为拟态界。拟态界的 DHR 构造和表决机制是拟态防御系统内生地具备安全能力的根源[15,18-19]。

为了防御基于二进制漏洞的进程控制流劫持攻击，本文受拟态防御思想的启发，采用动态异构冗余的方式执行程序。在异构冗余的基础上，结合表决机制，能够有效发现攻击行为，然后阻断攻击。

本文的主要研究工作及贡献如下。

1) 梳理了进程控制流劫持攻击的过程，从漏洞利用过程的角度建立了二进制漏洞威胁模型，并依据此模型提出了针对关键利用环节的“要塞”防御。

2) 提出了进程的拟态执行模型，为防御基于二进制漏洞的进程控制流劫持攻击提供了一种主动防御式的解决方案。

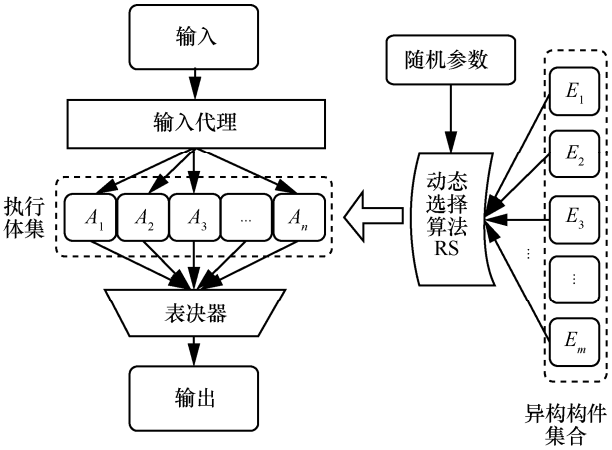

图 1 拟态防御系统 DHR 构造

3) 实现了一个拟态执行的原型系统—— MimicBox，并进行了有效性验证和性能测试。经测试，MimicBox 能够防御绝大多数已知的基于二进制漏洞的进程控制流劫持攻击，并且导致的额外性能开销不超过 13%。

2 二进制漏洞威胁模型

常见的控制流劫持技术往往是利用堆和栈开展攻击。在利用栈的攻击技术中[20]，栈缓冲区溢出攻击占了绝大部分。当栈缓冲区溢出，即程序接收的用户输入超过开发者所分配的缓冲区大小时，可能会破坏栈中所存储的关键数据，从而引发错误。由于栈中存放了函数的返回地址，因此在没有保护的情况下，一旦攻击者利用栈溢出，使用其他地址覆盖了原来的函数返回地址，那么当函数执行完成进行返回时，就会去执行攻击者所构造的地址处的代码，攻击者就可以达到控制流劫持的目的。在针对堆的攻击技术[21]中，攻击者首先利用申请和释放堆块时涉及的堆分配机制进行堆布局，然后非正常地申请和释放堆块，获得一个指向关键内存位置处的指针，最后使用程序的输入功能将其重写为其他代码段的地址，从而实现程序的控制流劫持。

本文分析了几种常见的面向 Linux 平台的控制流劫持方式，并以此为基础进行了二进制漏洞威胁建模，威胁模型如图 2 所示。模型遵循的前提与假设如下。

1) 目标机器装有 Linux 操作系统。

2) 攻击者拥有目标程序的可执行文件，能够对目标程序进行静态分析。

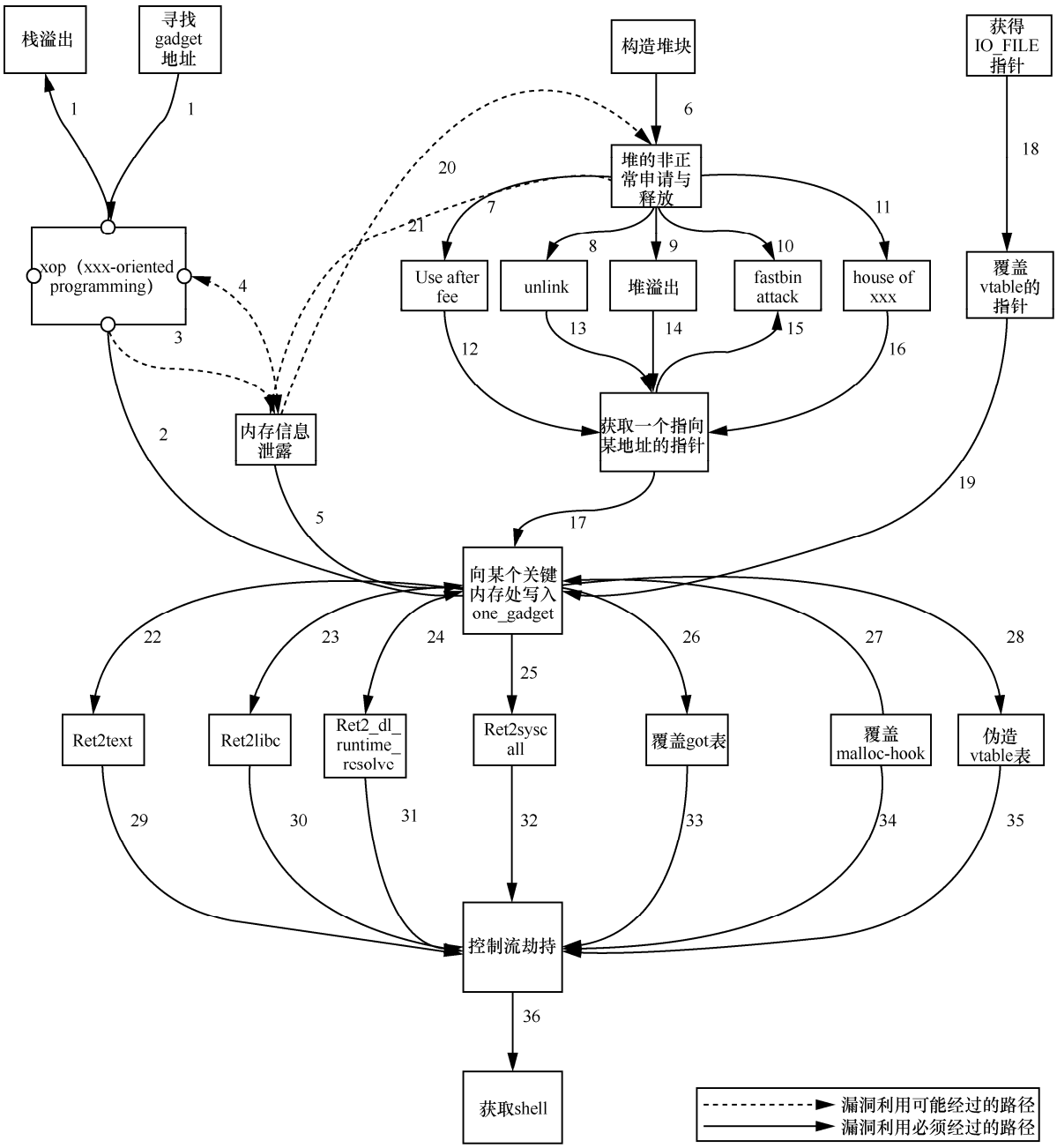

图 2　基于二进制漏洞的控制流劫持攻击威胁模型

3) 目标程序是个应用软件，并且包含可供控制流劫持使用的二进制漏洞。

4) 基于二进制漏洞的控制流劫持，目的在于获取目标机器控制权和 shell。

5) 将可以获取 shell 的代码片段统一记为 one_gadget。

在对栈溢出漏洞进行利用时，攻击者首先采取静态分析的方式获取 ROP 所需的 gadget 的地址信息，再以 ROP 技术作为跳板，实现程序的信息泄露，获取必要的内存信息，然后以此为基础，向栈中存放返回地址的内存处写入 one_gadget 地址，从而获取 shell，在某些特殊情况下，攻击者也可以使用字节爆破的方式在不获得信息的情况下向栈内注入恶意的地址。

在程序本身拥有可以实现控制流劫持的代码片段（如 system、execve 函数）的情况下，当目标程序被编译成地址无关可执行（PIE, position-independent executable）程序时，攻击者可以直

接将这些代码片段作为one_gadget，然后将栈中的返回地址覆盖为这些代码片段的地址，路径1—2—22—29—36表示一个完整的ret2text利用流程；当目标程序不是PIE程序时，攻击者将首先通过一个信息泄露漏洞，如格式化字符串漏洞，获取并计算出想要跳转到的one_gadget的地址，然后将栈中的返回地址覆盖为one_gadget的地址，路径4—2—22—29—36表示针对PIE程序的ret2text利用流程。

当程序本身并不拥有可以实现控制流劫持的代码片段时，攻击者往往将libc系统共享库中的system函数作为one_gadget，使“/bin/sh”作为system函数的参数即可。攻击者首先需要获取gadget地址信息，利用ROP等代码重用技术实现跳转完成信息泄露，然后获取libc库中某个函数的地址信息，通过偏移计算获得one_gadget的地址，最后再进行多次ROP，把got表中的某项篡改为one_gadget的地址，或者直接跳转到libc中的one_gadget。路径1—3—5—23—30—36表示一个完整的ret2libc利用流程。

路径1—2—25—32—36表示一个完整的ret2syscall攻击过程，攻击者使用ROP技术并在栈内注入syscall指令地址，然后劫持控制流获取shell。

路径6—7—12—17—26—33—36表示一个完整的释放后使用（UAF, use—after—free）漏洞利用流程。攻击者首先进行堆布局，对堆进行多次非正常申请与释放，获得一个指向got表中某表项的指针；然后攻击者利用程序自带的修改功能（如记事本程序中的编辑功能），把got表某项的指针指向的内容修改为one_gadget地址，从而完成程序的控制流劫持，获得shell。

路径6—8—13—17—27—34—36、6—9—14—17—27—34—36、6—10—15—17—27—34—36、6—11—16—17—27—34—36分别表示攻击者根据glibc管理堆的特性，采取不同的攻击方式的堆利用流程。其本质都是不断地构造特殊的堆并多次非正常地申请与释放堆，获得一个可以修改内容的指针，再使用程序自带的地址写功能将关键内存位置，如将got表中的表项或malloc_hook中的内容覆盖为one_gadget地址，从而达到控制流劫持获取shell的目的。

无论哪条漏洞利用路径，攻击者都无法摆脱对虚拟内存地址的依赖。在大部分情况下，攻击者首先通过信息泄露获取one_gadget的地址信息，然后向程序传入one_gadget地址，在其他情况下，攻击者可以跳过信息泄露这一步骤进行漏洞利用，但是仍然需要向程序注入正确的one_gadget地址。

3 拟态执行模型

要想实现一次完整的攻击过程，必须要知道one_gadget的虚拟内存地址。基于这一发现，本文认为防御的关键在于两点，一是不泄露one_gadget的地址，二是使泄露的one_gadget地址失效。因此，为了防御控制流劫持，本文提出拟态执行，将虚拟内存地址空间作为拟态界，在拟态界内部异构，当拟态界内与拟态界外产生数据交换时，则对数据进行表决。拟态执行模型如图3所示。

拟态执行过程如下。首先对一个即将运行的程序进行无规律的内存空间布局异构，异构成$n(n\geqslant 1)$份，并使之冗余执行。当有数据流出进程的虚拟内存空间时对数据内容进行表决，总共进行$m-1$（且n、m不能同时为1）次表决。当表决一致时，记录各冗余运行程序的状态；当表决不一致时，说明遭到了攻击，则阻断攻击并随之进行拟态变换，也就是再次进行无规律的变化运行时的内存空间布局。当$n=1$时，每次表决都触发一次拟态变换。

因为会对程序进行异构化处理，所以通过静态分析获得的one_gadget地址在实施攻击时是无效的。因此，针对拟态执行的攻击有2种情况。

情况1 爆破one_gadget地址，逐一尝试暴力枚举的one_gadget地址，直到成功。

情况2 如图4所示，通过2次漏洞利用完成，第一次上传payload利用程序本身的漏洞获取one_gadget地址，然后根据one_gadget再次构造并上传payload实施后面的攻击步骤，最终实现控制流劫持并获取shell，获取机器控制权。

下面，证明拟态执行具备防御控制流劫持的效果。记在拟态执行过程中触发的拟态变换次数为$t(m-1\geqslant t\geqslant 0)$。

情况1的证明如下。当$n=1$时，设寻址位数为k，在拟态变换的干扰作用下，一次地址爆破成功的概率始终为$p=\frac{1}{2^k}$，无论k取32还是64，都有$p\approx 0$；当$n>1$时，只可能按概率p爆破成功一路冗余执行进程中的one_gadget地址劫持控制流，然而，其他冗余进程仍按照原控制流执行，那么一定会在接下来的表决中发现不一致的拟态界内外数据交换，进而阻断攻击，然后进行拟态变换使

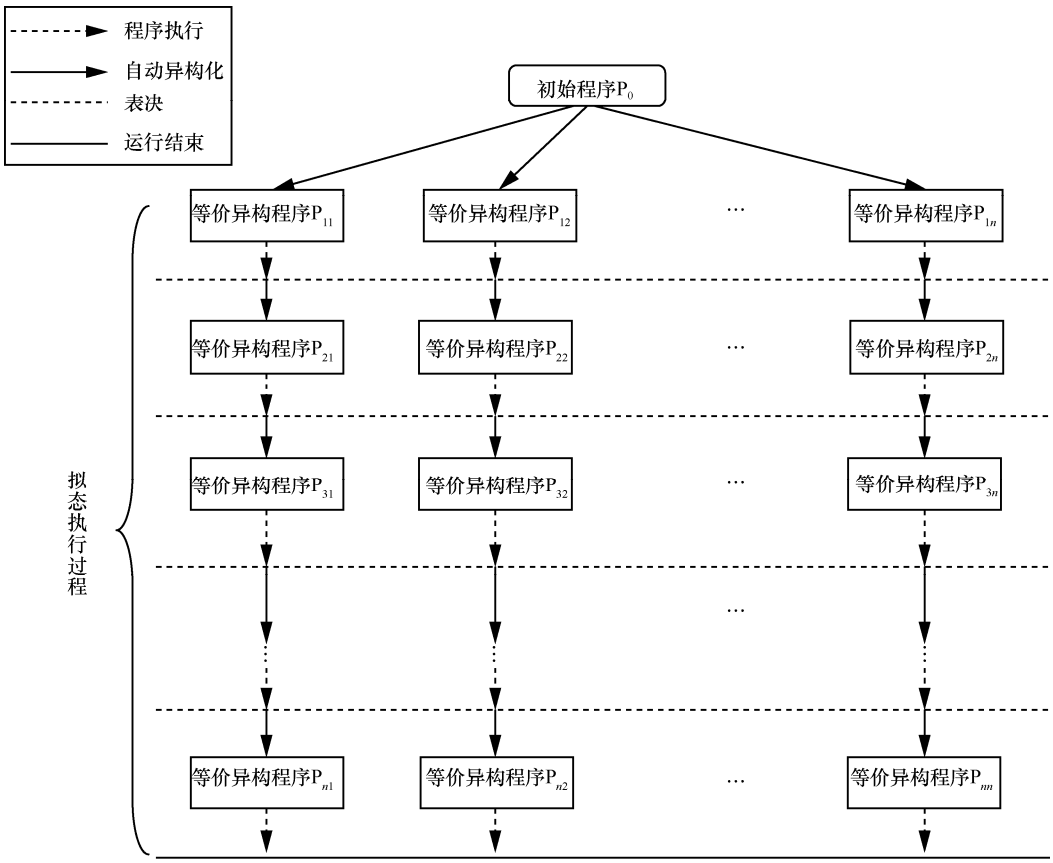

图 3　拟态执行模型

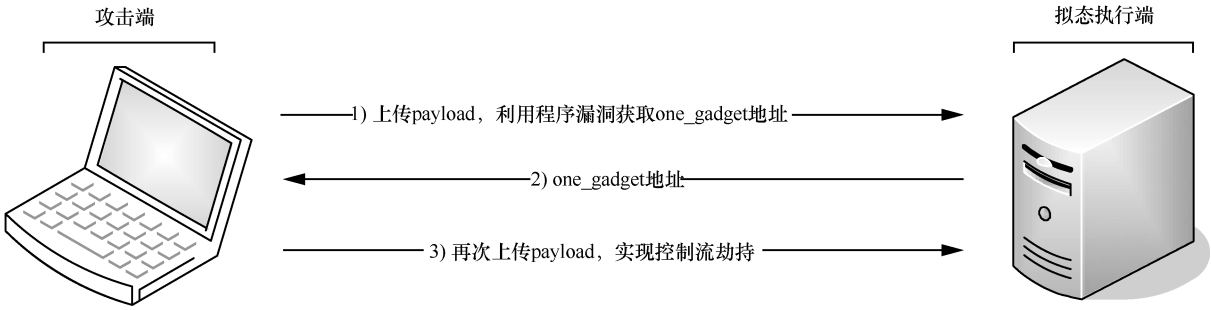

图 4　针对拟态执行的抽象攻击过程

one_gadget 地址失效。

情况 2 的证明如下。当 $n=1$ 时，每次数据流出拟态界时需进行表决，每次表决都会触发一次拟态变换，那么攻击者通过第一次上传 payload 获取的 one_gadget 地址在实施接下来的攻击步骤时（第二次上传 payload）无效。当 $n>1$ 时，攻击者通过第一次上传 payload 来泄露 one_gaaget 地址意味着 one_gadget 地址将作为数据流出拟态界，会触发一次表决，不同冗余执行程序中的 one_gadget 地址是不同的，那么在表决时一定会发现 one_gadget 地址泄露，从而阻断信息泄露，此时攻击端将无法进行接下来的攻击步骤。

4　原型系统实现

本文在 Linux 操作系统上实现了拟态执行的原型系统 MimicBox。如图 5 所示，MimicBox 是一个 $n>1$、$t=0$ 的拟态执行系统。MimicBox 对经异构处理的冗余执行的等价进程进行监控，当程序执行到表决点时进行表决，当表决判定为不一致时，MimicBox 将对程序进行阻断，从而保证 one_gadget 地址不泄露。

1) 进程异构化处理

Linux 操作系统本身就提供了许多内存空间布

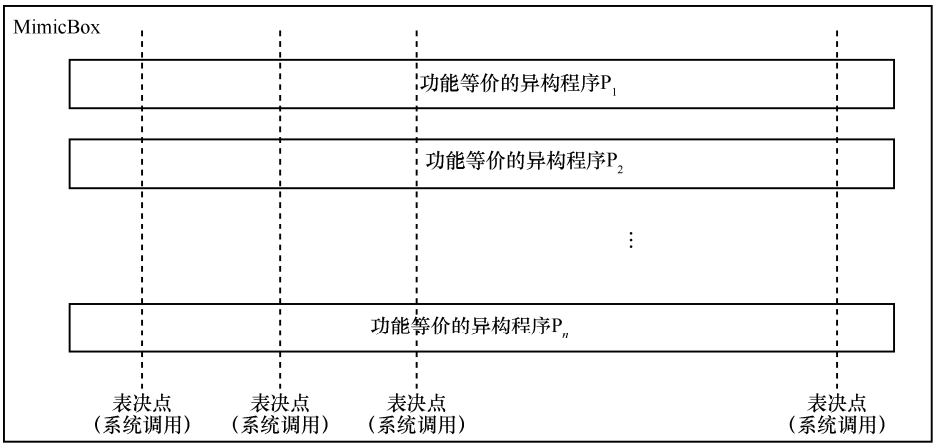

图 5　MimixBox 原理框架

局异构相关的技术，如 Canary 栈监控、PIE、ASLR 等。在编译时，Canary 技术通过在返回地址之前插入一个随机数，能够实现对程序代码本身的异构，PIE 程序在装载时可以被装载进随机的虚拟内存空间；在运行时，ASLR 会对执行进程的堆段、栈段、共享库的基地址进行随机化。ASLR 技术和 PIE 技术都能实现内存空间布局的异构。MimicBox 充分利用 Linux 操作系统提供的上述技术对程序进行异构化处理。

2) 表决点设置

因为拟态执行要求在拟态界内与拟态界外产生数据交换时进行表决，并且 MimicBox 的拟态界就是虚拟内存空间，所以当 MimicBox 发现冗余执行的进程有数据流出虚拟内存空间时，主动进行表决。因为 Linux 中一个进程的虚拟内存中的数据必须通过系统调用才能流向别处，所以 MimicBox 将表决点设置在所有输出类的系统调用。

3) 表决点监控

ptrace 系统调用是 Linux 操作系统向用户态提供的调试接口。当 ptrace 系统被调用时，会产生一个中断信号，ptrace 通过监听这个信号来判断是否产生系统调用。使用 ptrace 不仅能够监听系统调用的产生，而且还能控制进程的运行状态。MimicBox 使用 ptrace 系统调用实现系统调用的拦截、替换、表决内容提取、返回值覆盖等功能。

4) 冗余执行体管理

MimicBox 使用编号对冗余执行体进行管理，编号从 0 开始，依次递增 1。

5) 冗余输出合并

拟态界内部执行进程的冗余必将导致拟态界内与拟态界外数据交换的冗余，为了不对拟态界外造成影响，除了对数据交换进行表决之外，还必须合并冗余的数据交换。Linux 中使用文件描述符或网络套接字来标识具体的数据流向的目标，比如磁盘文件、socket 套接字等。MimicBox 使用目标点描述符来管理数据交换，目标点描述符的定义如图 6 所示。当 MimicBox 发现 open 或 openat 系统调用并且成功打开文件时，将创建一个目标点描述符。当 MimicBox 发现等价的冗余进程进行写操作时，就要检查目标点描述符的 desc 域，如果一致就进行合并。具体来说，使 0 号执行体中的进程正常执行，通过 getpid 系统调用替换其他等价进程中的系统调用，然后通过 0 号执行体中的进程系统调用的返回值覆盖其他等价进程中系统调用的返回值。另外，对于网络通信来说，除了要合并相关的输出类系统调用之外，还要对 connect、listen、accept、bind 等相关系统调用合并。合并方式与合并输出类系统调用的方式类似，不同的是，网络通信是在发现 connect、accept 系统调用时创建目标点描述符。

```
107  #define MAX_WIDTH     128
108
109  typedef struct dest_point
110  {
111      int fd;
112      int type;                // local or net
113      char desc[MAX_WIDTH];
114  } d_fd_t;
```

图 6　目标点描述符的定义

6) 等价线/进程匹配

为了更好地支持多线/进程，在表决时要准确定位到相互功能等价的冗余线/进程，这要求等价线/进程在 MimicBox 中要有相同的标识。在 MimicBox 中，对线/进程按照层次进行编号，使用一系列编号

组成的序列标识线/进程。图 7 展示了冗余等价任务中其中一个任务的线/进程关系，进程 pid=2 646 由进程 pid=2 645 通过调用 fork 函数得到，一个线程从 0 开始给它的子线程编号，当再创建新线程时，编号依次递增 1。对于一个线程，沿着父线程方向回溯，直到进程的第一个线程，由所途经线程的编号组成的序列即为这个线程的标识。以线程 tid=2 660 和线程 tid=2 654 为例，它们在该执行体中的唯一标识序列分别为(2,0,1)和(0,1,0)。在冗余等价任务中，具有相同标识的线程视为等价线程。

5　评估

本文从防御有效性和性能 2 个方面对 MimicBox 进行了实验评估。实验旨在回答如下 2 个问题。

1) MimicBox 的防御边界在哪里？能防御哪些类型的攻击？

2) 任何安全措施必将导致效率的降低，MimicBox 是否会带来不可接受的性能损耗？

5.1　有效性验证和评估

为了验证 MimicBox 防御控制流劫持攻击的有效性，设置了 3 种测试环境，并在 3 种不同的场景进行有效性验证实验或测试。

5.1.1　理想场景验证实验

理想环境下的验证实验使用 CTF Pwn 夺旗赛中的题目作为实验样本。在 CTF Pwn 夺旗赛中，通常会将编译好的带漏洞可执行程序发放给选手，选手通过对二进制程序进行逆向分析和调试来找出漏洞，并编写攻击代码，以远程代码执行的方式拿到 flag。

Pwn 程序中的漏洞覆盖面广泛，几乎包含了绝大多数已知的二进制漏洞。一般来说，Pwn 程序中除了预先设置的漏洞之外不包含其他漏洞，能使实验更有针对性。因此本文采用 CTF wiki 上的题目作为实验样本，进行验证实验。

使用 MimicBox 将 CTF wiki 中的 Pwn 题目对应的带漏洞程序保护起来，然后用 CTF wiki 中提供的 expliot 脚本进行远程攻击。每次执行 exploit 脚本记录下是否获取目标机器的 shell，获取不到 shell 则意味着攻击失败，防御成功。攻击实验结果如表 1 所示。对于栈攻击、堆攻击和 I/O 文件攻击，MimicBox 均能防御相关的漏洞利用。

5.1.2　真实攻防场景测试

为了测试在真实攻防场景下 MimicBox 的有效防御能力，将 MimicBox 放到了第三届“强网”拟态防御国际精英挑战赛[22]，接受来自世界上 40 支战队的挑战。第三届“强网”拟态防御国际精英挑战赛在 2020 年 6 月 19 日举行，为期两天。此次比赛按照“BMW”模式[23]，采取线上白盒积分争夺赛、线上拟态白盒挑战赛、线上拟态黑盒挑战赛的全新赛制。其中，白盒积分争夺赛采取 CTF Pwn 比赛的形式与规则，MimicBox 出现在白盒积分争夺赛上。

比赛共出了四道题目，其中两道题目中的程序未被 MimicBox 保护，作为基础题；另外两道题目中将同样的带漏洞程序采用 MimicBox 保护，作为进阶题。经过 48 h 的不间断攻击挑战，完成了对比测试。

题目 easy-stack 中的目标程序包含一个格式化字符串漏洞和一个栈溢出漏洞；题目 newpad 的对应程序中包含一个 House of einherjar 类型的堆漏洞。经过测试，大部分战队能解出基础题目，然而却没有一支战队能解出进阶题目，这意味着没有任

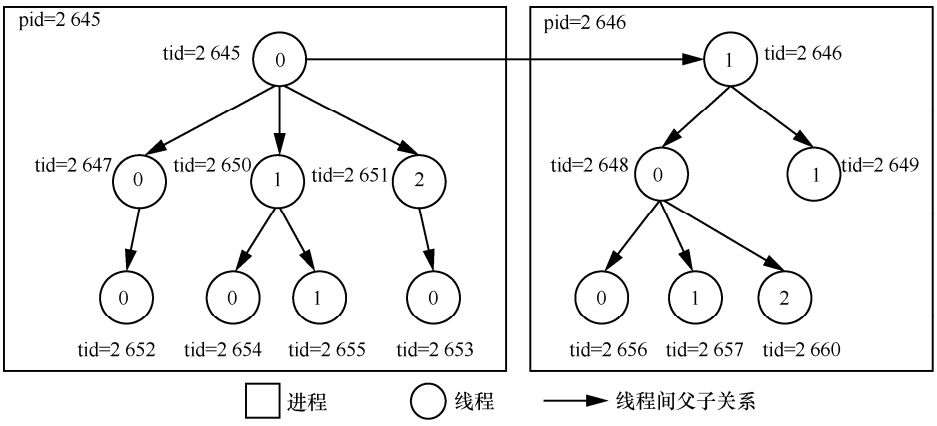

图 7　MimicBox 中的进/线程关系

表 1 **CTF Pwn 测试结果**

攻击类型	利用方式	Pwn 题目名称	题目出处	异构方式	冗余度	能否防御
栈攻击	ret2text	ret2text	Bamboofox ctf	ASLR+PIE	2	√
	ret2libc	ret2libc	Bamboofox ctf	ASLR+Canary	2	√
	ret2syscall	ret2syscall	Bamboofoxv ctf	ASLR+PIE	2	√
	Ret2_dll_runtime_resolve	Pwn200	XDCTF 2015	ASLR+PIE+Canary	2	√
堆攻击	Use after free	Lab 10 hacknote	HITCON-training	ASLR+PIE	2	√
	堆溢出	stkof	2014 HITCON	ASLR+PIE	2	√
	Unlink attack	Note2	2016 ZCTF	ASLR+PIE	2	√
	Fastbin attack	Oreo	2014 hack.lu	ASLR+PIE	2	√
	House of einherjar	Tinypad	2016 Seccon	ASLR+PIE	2	√
	House of Roman		Romanking98	ASLR+PIE	2	√
	House of force	Bcloud	2016 BCTF	ASLR+PIE	2	√
	House of orange		CTF wiki	ASLR+PIE	2	√
	House of lore		CTF wiki	ASLR+PIE	2	√
	House of rabbit		CTF wiki	ASLR+PIE	2	√
I/O 文件攻击	Vtable 劫持	Pwn450	东华杯 2016	ASLR+PIE	2	√

何战队能够突破 MimicBox 的防护，具体测试结果如表 2 所示。

5.1.3 真实漏洞利用场景验证实验

理想场景实验和真实攻防场景测试表明，MimicBox 对一些基础漏洞完全具备防御能力，为了验证 MimicBox 对信息泄露的防御能力，选取了 2 个真实世界中的软件漏洞（CVE-2014-0160 和 CVE-2018-16890）进行漏洞防护实验。

CVE-2014-0160 是 OpenSSL 中的一个高危漏洞，源于 OpenSSL 中的 TSL 模块和 DTLS 模块在调用 memcpy 函数之前没有对长度参数进行边界检查，将导致过多的内存信息复制到缓冲区以每次 64 KB 的速度泄露，泄露信息可能包括服务器私钥、用户 cookie 和密码等。

MimicBox 能够有效防御 CVE-2014-0160。当内存信息向外泄露时，将会触发表决。由于对冗余执行的等价进程做了异构处理，各冗余进程泄露的内存信息不一致，MimicBox 表决时将发现不一致，从而阻断信息泄露。

CVE-2018-16890 是 7.36.0 版本到 7.64.0 版本的 curl 中的一个整数溢出漏洞。curl 处理 NTML type-2 消息的函数 ntlm_decode_type2_target 没有验证数据是否正确，利用整数溢出，恶意代码或者损坏的 NTLM 服务器能够使 curl 接收到错误的数据，将导致对一个堆缓冲区的越界读取。利用此漏洞，攻击者可以在服务器上远程获取客户端内存至多 64 KB 的原始内存信息。而且因为连接可以多次进行，服务器理论上可以多次重复地获取客户端内存信息。

使用 MimicBox 保护 curl 客户端，能够有效防止信息泄露。防御原理与防御 CVE-2018-16890 类似，MimicBox 将通过表决发现各冗余进程泄露的不一致信息从而进行阻断，具体来说，其发现了 sendto 系统调用向服务端发送的内容不一致。当客户端被 MimicBox 保护后，在服务端接收不到客户端泄露的内存信息。

表 2 **挑战赛测试结果**

题目名称	题目类型	包含漏洞	异构方法	MimicBox 保护	突破队伍数目
easy-stack	基础	格式化字符串、栈溢出	ASLR+PIE+Canary	否	38
newpad	基础	House of einherjar	ASLR+PIE+Canary	否	32
Advanced easy-stack	进阶	格式化字符串、栈溢出	ASLR+PIE+Canary	是	0
Advanced newpad	进阶	House of einherjar	ASLR+PIE+Canary	是	0

5.1.4　有效性评估

理想环境下的验证实验结果表明，MimicBox 能够防御大部分栈漏洞、堆漏洞和 I/O 文件漏洞利用。另外，如表 1 所示，在有效异构的前提下，MimicBox 只需要使用双余度即可保证有效性。

真实攻防环境下的测试表明，相对于单一的基于随机的异构结构，MimicBox 的异构冗余结构赋予了程序极强的内生安全能力。

真实漏洞利用场景下的验证实验表明，对于以心脏滴血为代表的信息泄露类漏洞来说，拟态执行中蕴含的异构冗余特性使 MimicBox 具备天然的防御能力。

总而言之，经过 3 种不同测试场景下的充分实验，对于第一个问题，拟态执行的原型系统 MimicBox 能够有效阻断绝大部分已知的基于栈利用、堆利用、I/O 文件攻击和信息泄露的进程控制流劫持。

5.2　性能评估

针对 MimicBox 的性能测试，则使用 SPEC CPU2006 完成。具体实验环境配置如表 3 所示。首先对 SPECint 2006 中的 8 个样本进行基准测试，然后用 MimicBox 将这 8 个样本保护起来，测试多余度情况下的性能并计算额外性能开销（overhead）。测试结果如图 8 所示，额外性能开销结果如表 4 所示。

对比基准程序和 MimicBox 多余度测试结果，MimicBox 所需的时间、额外性能开销与冗余度呈正相关关系。横向观察表 4，随着冗余度的增加，额外性能开销也随之增加。纵向观察表 4，当 n=2 时，所有样本的额外性能损耗都不超过 13%。对于 CPU 密集型程序来说，MimicBox 导致的性能损耗是可以接受的。

表 3　　性能测试环境配置

条目	参数
CPU 型号	Intel(R) Xeon® CPU E5-2603 v4@ 1.70 GHz 6 cores
内存大小	32 GB
操作系统	CentOS 7.3
Linux kernel 版本	3.10.0-1127.10.1.el7.x86_64

5.3　对比评估

在已知的面向进程控制流劫持的防御方法中，CFI 是较有效的一种。本节将对拟态执行与 CFI 进行一个简单的对比。

CFI 认为只有符合控制流图的控制流转移才是合法的[24]。根据输入程序类型的不同，控制流完整性实施方案可分为面向源代码和面向二进制 2 种[25]；根据控制流转移时检查的精细程度，CFI 又分为粗粒度 CFI 和细粒度 CFI[12]。CFI 被认为是较有效的一种防御控制流劫持的方案，然而，CFI 仍然存在一些局限性。

粗粒度的 CFI 往往有效性较差，而细粒度的 CFI 为了追求高有效性，往往导致不可接受的额外性能开销。文献[26]中的 Picon 是一个面向源代码

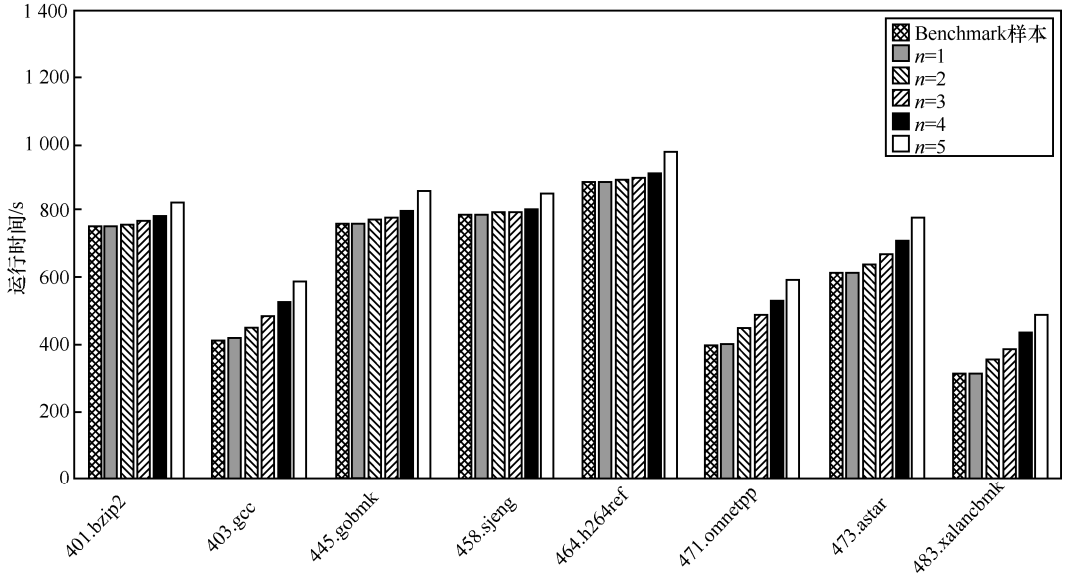

图 8　测试结果

表 4 MimicBox 多余度额外性能损耗

Benchmark 样本	*n*=1	*n*=2	*n*=3	*n*=4	*n*=5
401.bzip2	0.06%	1.03%	2.18%	4.34%	10.11%
403.gcc	1.08%	8.17%	16.78%	26.87%	41.57%
445.gobmk	0.09%	1.39%	2.50%	5.02%	13.19%
458.sjeng	0.08%	0.70%	1.17%	2.20%	7.75%
464.h264ref	0.07%	0.89%	1.25%	3.03%	10.52%
471.omnetpp	0.79%	12.51%	22.27%	33.67%	49.48%
473.astar	0.17%	4.35%	8.72%	15.32%	26.50%
483.xalancbmk	0.79%	12.36%	24.58%	38.57%	57.52%

的 CFI 框架，导致程序的平均运行时间增加了 900 多倍；Bincon[27]是一个面向二进制的 CFI 框架，导致程序的平均运行时间增加了 20 多倍。为了同时追求高有效性和低开销，一些研究者着手于硬件加速，以现有的硬件手段来降低 CFI 的开销[25-30]。使用硬件虽然能带来一定的开销下降，但是却不可避免地带来了一定的成本上升。

拟态执行却是一种轻量、高有效、低开销、低成本的防御方案。拟态执行不需要在程序执行之前建立控制流图，具备内生安全能力，额外性能开销较低，成本低廉。拟态执行与面向源代码的 CFI 相比，对源代码的依赖程度小；与面向二进制的 CFI 相比，不需要进行反编译或插桩；与粗粒度的 CFI 相比，能够防御的攻击类型更广泛；与细粒度的 CFI 相比，有效防御能力相当，但是拟态执行的额外性能开销远远小于 CFI。

另外，CFI 无法抵御内存信息泄露，而拟态执行能够有效防止内存信息泄露。

总之，拟态执行是相对于 CFI 更实用的一种用来防御进程控制流劫持的方法。

5.4 局限性

即使 MimicBox 是退化的软件拟态执行系统原型，但是经过实验验证，其几乎可以防御绝大部分已知类型的控制流劫持攻击。然而，MimicBox 仍存在以下 2 个方面的局限性。

1) 对于 I/O 密集型程序，MimicBox 将可能导致较大的性能下降。性能下降的主要原因一方面是拟态系统的内在特性，另一方面是过多的无意义的系统调用处理。拟态执行强调在拟态界内向拟态界外流出数据时进行表决，而 MimicBox 使用 ptrace 拦截所有的系统调用，对其中的部分系统调用进行表决，这大大降低了 ptrace 的有效使用率。另外，使用 ptrace 提取系统调用参数、控制进程状态等操作使整个系统中多出许多次额外的系统调用。

2) 存在表决假阳导致的误报问题。导致表决假阳的主要原因是 MimicBox 中冗余进程之间存在内生的不一致属性，比如进程号、随机数、文件描述符等。当这些不一致属性作为数据流出拟态界时，往往导致表决误报。

6 结束语

为了防御进程控制流劫持攻击，本文提出了拟态执行方法。将拟态防御的 DHR 构造引入进程执行过程中，在异构冗余的基础之上进行表决，能够有效发现攻击并采取防御措施。

拟态执行的原型系统 MimicBox 验证了拟态执行的可行性和防御有效性，同时说明拟态执行具有良好的发展前景。拟态执行是一种轻量的、对用户透明的防御方式，应用领域可以涵盖云安全、系统安全、物联网设备安全等。

本文建立了进程控制流劫持攻击的威胁模型，经过对模型的分析，提出了进程的拟态执行防御模型，并基于该模型，实现了原型系统。经过对原型系统的安全性测试，验证了拟态执行的防御有效性。最后提出了拟态执行的发展前景和推广价值。

参考文献：

[1] COWAN C, WAGLE P, PU C, et al. Buffer overflows: attacks and defenses for the vulnerability of the decade[C]//DARPA Information Survivablity Conference and Exception. Piscataway: IEEE Press, 2000: 119-129.

[2] 王丰峰, 张涛, 徐伟光, 等. 进程控制流劫持攻击与防御技术综述[J]. 信息安全学报, 2019, 5(6):10-20.
WANG F F, ZHANG T, XU W G, et al. Overview of control-flow hijacking attack and defense techniques for process[J]. Chinese Journal of Network and Information Security, 2019, 5(6): 10-20.

[3] MITRE. 2020 CWE top 25 most dangerous software errors[R]. (2020-08-20)[2020-08-26].

[4] VEN A. New security enhancements in red hat enterprise linux v.3, update 3[R]. 2004.

[5] COWAN C, PU C, MAIER D, et al. Stackguard: automatic adaptive

detection and prevention of buffer-overflow attacks[J]. Usenix Security, 1998, 98: 63-78.
[6] PaX Team. PaX ASLR[R]. 2003.
[7] ROEMER R, BUCHANAN E, SHACHAM H, et al. Return-oriented programming: systems, languages, and applications[J]. ACM Transactions on Information and System Security, 2012, 15(1): 1-34.
[8] 乔向东, 郭戎潇, 赵勇. 代码复用对抗技术研究进展[J]. 网络与信息安全学报, 2018, 4(3): 1-12.
QIAO X D, GUO R X, ZHAO Y. Research progress in code reuse attacking and defending[J]. Chinese Journal of Network and Information Security, 2018, 4(3):1-12.
[9] 邢骁, 陈平, 丁文彪, 等. BIOP: 自动构造增强型 ROP 攻击[J]. 计算机学报, 2014, 37(5): 1111-1123.
XING X, CHENG P, DING W B, et al. BIOP: automatic construction of enhanced ROP attack[J]. Chinese Journal of Computers, 2014, 37(5): 1111-1123.
[10] 陈振伟, 孙歆. 使用 ROP 技术突破 Linux 的 NX 防护研究[J]. 网络空间安全, 2018, 9(2): 64-69.
CHEN Z W, SUN X. Research on bypassing the NX protection of Linux with ROP[J]. Cyberspace Security, 2018, 9(2): 64-69.
[11] EVTYUSHKIN D, PONOMAREV D, ABU-GHAZALEH N. Jump over ASLR: attacking branch predictors to bypass ASLR[C]//IEEE/ACM International Symposium on Microarchitecture. New York: ACM Press, 2016: 1-13.
[12] 武成岗, 李建军. 控制流完整性的发展历程[J]. 中国教育网络, 2016(4): 52-55.
WU C C, LI J J. The evolution of control flow integrity[J]. China Education Network, 2016(4): 52-55.
[13] SAYEED S, MARCO-GISBERT H. On the effectiveness of control-flow integrity against modern attack techniques[M]. Berlin: Springer, 2019.
[14] HU H, SHINDE S, ADRIAN S, et al. Data-oriented programming: on the expressiveness of non-control data attacks[C]//2016 IEEE Symposium on Security and Privacy. Piscataway: IEEE Press, 2016: 969-986.
[15] 邬江兴. 网咯空间内生安全——拟态防御与广义鲁棒控制（上册）[M]. 北京: 科学出版社, 2020.
WU J X. Endogenous security of cyber space:mimic defense and generalized robust control (volume 1)[M]. Beijing: Science Press, 2020.
[16] 邬江兴. 网咯空间内生安全——拟态防御与广义鲁棒控制（下册）[M]. 北京: 科学出版社, 2020.
WU J X. Endogenous security of cyber space:mimic defense and generalized robust control (volume 2)[M]. Beijing: Science Press, 2020.
[17] 邬江兴. 网络空间拟态防御研究[J]. 信息安全学报, 2016, 1(4): 1-10.
WU J X. Research on cyber mimic defense[J]. Journal of Cyber Security, 2016, 1(4): 1-10.
[18] 仝青, 张铮, 张为华, 等. 拟态防御 Web 服务器设计与实现[J]. 软件学报, 2017, 28(4): 883-897.
TONG Q, ZHANG Z, ZHANG W H, et al. Design and implemention of mimic defense Web server[J]. Journal of Software, 2017, 28(4): 883-897.
[19] 张铮, 马博林, 邬江兴. Web 服务器拟态防御原理验证系统测试与分析[J]. 信息安全学报, 2016, 2(1): 13-28.
ZHANG Z, MA B L, WU J X. The test and analysis of prototype of mimic defense in Web servers[J]. Journal of Cyber Security, 2017, 2(1): 13-28.
[20] 曾永瑞, 李喆. Linux 二进制漏洞利用——突破系统防御的关键技术[J]. 信息安全研究, 2018, 4(9): 806-818.
ZENG Y R, LI Z. Linux binary exploit——The key technology of breaking through the system defense[J]. Journal of Information Security Research, 2018, 4(9): 806-818.
[21] 裴中煜, 张超, 段海新. Glibc 堆利用的若干方法[J]. 信息安全学报, 2018, 3(1): 1-15.
PEI Z Y, ZHANG C, DUAN H X. Serval methods of exploiting glic heap[J]. Journal of Cyber Security, 2018, 3(1): 1-15.
[22] 第三届“强网”拟态防御国际精英挑战赛在南京开幕[N]. 新华网, 2020-06-19.
The 3rd “strong net” mimic defense international elite challenge opens in Nanjing[N]. News, 2020-06-19.
[23] 邬江兴. 加快推进网络安全学科竞赛创新发展[N]. 解放军报, 2019-07-05.
WU J X. Accelerate the innovation and development of cybersecurity discripline competition[N]. PLA Daily, 2019-07-05.
[24] MARTN A. Control-flow integrity[C]//Proceedings of the 12th ACM Conference on Computer and Communications Security. New York: ACM Press, 2005: 340-353.
[25] PAPPAS V, POLYCHRONAKIS M, KEROMYTIS A D. Transparent ROP exploit mitigation using indirect branch tracing[C]//Usenix Conference on Security. Berkeley: USENIX Association, 2013: 447-462.
[26] COUDRAY T, FONTAINE A, CHIFFLIER P. PICON: control flow integrity on LLVM IR[C]//Symposium on Security of Information on and Communication Technology.[S.n.:s.l.], 2015: 1-6.
[27] 帕尔哈提江·斯迪克, 马建峰, 孙聪. 一种面向二进制的细粒度控制流完整性方法[J]. 计算机科学, 2019, 46(S2): 417-420, 432.
SIDIKE PA-ER H T J, MA J F, SUN C. Fine-graine control flow integrity method on binaries[J]. Computer Science, 2019, 46(S2): 417-420, 432.
[28] CHENG Y, ZHOU Z, MIAO Y, et al. ROPecker: a generic and practical approach for defending against ROP attack[C]//Network and Distributed System Security Symposium. [S.n.: s.l.], 2014: 1-14.
[29] FENG L, HUANG J, HU J, et al. FastCFI: real-time control flow integrity using FPGA without code instrumentation[C]//International Conference on Runtime Verification. Berlin: Springer, 2019: 221-238.
[30] KAWADA T, HONDA S, MATSUBARA Y, et al. TZmCFI: RTOS-aware control-flow integrity using trustzone for Armv8-M[J]. International Journal of Parallel Programming, 2020, doi:10.1007/s10766-020-00673-z.

[作者简介]

潘传幸（1996− ），男，山东梁山人，信息工程大学博士生，主要研究方向为主动防御、拟态防御等。

张铮（1975− ），男，湖北黄冈人，博士，信息工程大学教授，主要研究方向为高性能计算、拟态防御等。

马博林（1993− ），男，山东青岛人，信息工程大学博士生，主要研究方向为主动防御、多变体执行技术、拟态防御等。

姚远（1972− ），男，湖北武汉人，博士，信息工程大学教授，主要研究方向为先进计算、并行处理等。

季新生（1968− ），男，江苏南通人，博士，国家数字交换系统工程技术研究中心教授、博士生导师，主要研究方向为移动通信网络、拟态安全等。