

面向进程多变体软件系统的攻击面定性建模分析

邢福康¹, 张铮¹, 隋然², 曲晟¹, 季新生¹

(1. 信息工程大学, 河南 郑州 450001; 2. 中央军委后勤保障部信息中心, 北京 100089)

摘要: 攻击面是衡量软件系统安全性的一个重要指标, 采用攻击面描述可以通过集合的方式描述软件系统的安全性并对其进行度量。一般的攻击面模型基于 I/O 自动机模型对软件系统进行建模, 其一般采用非冗余的架构, 难以应用于类似多变体系统这类异构冗余的系统架构。Manadhatad 等提出了一种在非相似冗余系统中进行攻击面度量的方式, 但其采用的系统架构表决粒度和表决方式与多变体系统不同, 无法准确度量多变体系统的攻击面。因此, 在传统攻击面模型基础上, 结合多变体系统异构冗余架构的特点, 对传统攻击面模型进行扩展, 并构建多变体系统的攻击面模型; 使用形式化方式表示多变体系统的攻击面, 根据多变体系统在系统出口点处的表决机制对传统攻击面模型进行改进, 以使其能解释多变体系统攻击面缩小的现象, 通过该建模方式, 能够说明采用多变体架构的多变体系统在运行过程中攻击面的变化。采用了两组多变体执行架构的软件系统进行实例分析, 分别通过与未采用多变体架构的功能相同的软件系统在未受攻击和遭受攻击两种情境下进行攻击面的对比分析, 体现多变体系统在攻击面上的变化。结合攻击面理论与多变体执行系统的特点提出了一种面向多变体执行系统的攻击面建模方法, 目前可以定性分析多变体执行系统攻击面的变化, 未来将在定量分析多变体执行系统攻击面的方向继续进行深入研究。

关键词: 多变体; 攻击面; 攻击面度量; 网络安全

中图分类号: TP393

文献标志码: A

DOI: 10.11959/j.issn.2096-109x.2022059

Qualitative modeling and analysis of attack surface for process multi-variant execution software system

XING Fukang¹, ZHANG Zheng¹, SUI Ran², QU Sheng¹, JI Xinsheng¹

1. Information Engineering University, Zhengzhou 450001, China

2. Information Center of Logistics Support Department of Central Military Commission, Beijing 100089, China

Abstract: Attack surface is an important index to measure security of software system. The general attack surface model is based on the I/O automata model to model the software system, which generally uses a non-redundant ar-

收稿日期: 2022-01-18; 修回日期: 2022-04-25

通信作者: 张铮, ponyzhang@126.com

基金项目: 国家自然科学基金(61521003); 国家重点研发计划(2018YF0804003, 2017YFB0803204)

Foundation Items: The National Natural Science Foundation of China (61521003), The National Key R&D Program of China (2018YF0804003, 2017YFB0803204)

引用格式: 邢福康, 张铮, 隋然, 等. 面向进程多变体软件系统的攻击面定性建模分析[J]. 网络与信息安全学报, 2022, 8(5): 121-128.

Citation Format: XING F K, ZHANG Z, SUI R, et al. Qualitative modeling and analysis of attack surface for process multi-variant execution software system[J]. Chinese Journal of Network and Information Security, 2022, 8(5): 121-128.

chitecture and it is difficult to apply to heterogeneous redundant system architectures such as multi variant systems. Manadhatad et al. proposed a method to measure the attack surface in a dissimilar redundancy system. However, the voting granularity and voting method of the system architecture adopted by Manadhatad are different from those of the multi-variant system, which cannot accurately measure the attack surface of the multi variant system. Therefore, based on the traditional attack surface model, combined with the characteristics of heterogeneous redundant architecture of multi variant systems, the traditional attack surface model was extended and the attack surface model of multivariant systems was constructed. The attack surface of the multi variant system was represented in a formal way, and the traditional attack surface model was improved according to the voting mechanism of the multi variant system at the exit point of the system, so that it can explain the phenomenon that the attack surface of the multi variant system shrinks. Through this modeling method, the change of the attack surface of the multi variant system adopting the multi variant architecture can be explained in the running process. Then, two groups of software systems with multi variant execution architecture were used as analyzing examples. The attack surface of the software systems with the same functions as those without multi variant architecture were compared and analyzed in two situations of being attacked and not being attacked, reflecting the changes of the multi variant system in the attack surface. Combining the attack surface theory and the characteristics of the multi variant execution system, an attack surface modeling method for the multi variant execution system was proposed. At present, the changes of the attack surface of the multi variant execution system can be qualitatively analyzed. In-depth research in the quantitative analysis of the attack surface of the multi variant execution system will be continually conducted.

Keywords: multi-variant execution, attack surface, attack surface metric, network security

0 引言

随着信息技术的高速发展,越来越多的软件系统出现,软件系统服务于社会的各个层面,为人们的日常生活工作提供了极大的便利。然而,随着软件系统深入人们的生活,软件系统的功能和保存的数据信息越来越重要,针对软件系统进行的攻击日益增多,保护软件安全已成为保护网络空间安全^[1]的重要一环。1988年,Morries 利用软件系统的缓冲区溢出漏洞破坏了大量的计算机^[2]。近年来,随着软件系统的发展,软件漏洞层出不穷,如 Bletsch 等^[3]提出了面向跳转的攻击,Levy^[4]提出了代码注入攻击,而目前常见的攻击方式为面向返回的攻击,由 Shacham 于 2007 年提出^[5]。

为了应对这些层出不穷的攻击行为,很多相应的防御措施被提出。这些防御措施依据引入时间的不同被分为运行前的静态防御以及运行时的动态防御^[6]。静态防御方式,如 StackGuard^[7]在 gcc 编译时增加 canary 机制防止攻击。Cowan 等^[8]通过在编译时替换易受攻击的库进行软件的静态防护。但是,由于 Bittau 等提出的无源码的面向

返回攻击^[9]可以在不依赖源码的情况下获得软件的敏感信息,从而进行控制流劫持,静态防御措施的作用越来越小。在动态防御的措施中,控制流完整性技术^[10]通过检测程序的执行流程是否被篡改来防御攻击,然而由于其性能较差,目前阶段较难应用于实际生产中的软件系统。马博林等^[11]提出了基于指令集随机化的抗代码注入攻击方法,该方法基于指令集随机化防御未知代码注入攻击。多变体执行技术由 Cox^[12]提出,其根据两个功能相同的进程在内存分布上的随机化以及监控器保证软件运行的安全性。在此基础上,Salamat^[13]提出了监控器独立运行的多变体架构,而 Koning 等^[14]提出了 MvArmor 多变体系统架构,该架构基于对系统调用进行表决,采用该架构的用户可以在安全性和性能两个维度上进行选择,以在保护软件安全性的前提下保证其性能。

攻击面是衡量软件系统安全性的一个重要指标,攻击面描述可以通过集合的方式来描述软件系统的安全性并对其进行度量。传统的攻击面模型基于 I/O 自动机模型对软件系统进行建模,其一般采用非冗余的架构,难以应用于类似多变体系统这类异构冗余的系统架构。文献[15]提出了

一种在非相似冗余系统中进行攻击面度量的方式，但其进行攻击面建模的系统架构在异构冗余粒度、表决粒度等方面与多变体系统不同。该方式针对的系统为异构冗余架构的 Web 应用系统，一般通过服务器软件、应用层脚本等 Web 应用的组件的异构冗余提升系统的安全性，并针对异构组件产生的输出进行表决。进程多变体软件系统主要针对进程的内存空间进行异构冗余，同时对进程运行中产生的系统调用进行表决，其异构冗余粒度和表决粒度要远小于异构冗余架构的 Web 应用系统，并且其表决方式与异构冗余架构的 Web 应用系统有较大差异，因此文献[15]提出的方式无法准确度量多变体系统的攻击面。本文在传统攻击面模型的基础上，提出一种能够描述多变体系统攻击面的攻击面建模方式，从而说明采用多变体架构的多变体系统在运行过程中攻击面的变化，并通过实际的攻击展示多变体系统攻击面的变化。

1 相关工作

1.1 多变体系统

与传统的软件系统相比，多变体系统的设计引入了异构冗余的思想，其利用多个功能等价但存在异构性的变体进程对使用者透明地提供与传统的软件系统相同的服务。由于变体之间存在内存空间的异构性，当攻击者进行攻击时，很难对全部变体攻击成功，对变体执行流程中的系统调用进行监控，能够对系统中各个变体执行流程中表现出的不一致进行表决，从而阻止攻击者的攻击行为。而传统的软件系统由于没有异构冗余，无法进行执行流程的表决，更易被攻击者攻击成功。图 1 为典型的多变体系统架构模型。

多变体系统的核心模块主要包括输入模块、多变体模块和监控表决模块。各模块的作用如下。

输入模块：将系统输入进行分发，交给各个多变体进程进行执行。

多变体模块：含有多个功能等价的进程，进程均采用地址空间配置随机加载（ASLR）、地址无关可执行文件（PIE）等地址随机化技术，使得各个进程的内存分布不相同。

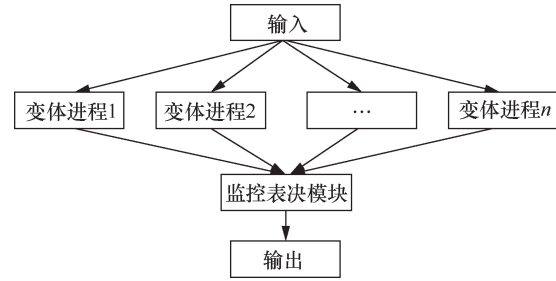


图1 典型的多变体系统架构模型
Figure 1 Architecture model of typical multi-variant execution software system

监控表决模块：监控每个进程的执行，在进程需要调用系统时进行拦截，比较各个多变体进程的内存内容和系统是否一致，若一致则继续执行，否则视为监测到攻击，结束多变体系统的执行。

文献[16]结合动态、异构、冗余的拟态防御思想，提出能够防御进程控制流劫持攻击的多变体系统 MimicBox，文献[17]通过传统数据库系统中引入多变体架构和数据库代理表决，实现了数据库系统对于 SQL 注入攻击的运行时防御。

1.2 攻击面理论

传统攻击面模型由 Manadhata 等^[18]提出，在传统攻击面模型中，攻击面被定义为可被攻击者利用的系统资源的总和。Manadhata 将系统攻击面的度量划分为 3 个维度，即系统的方法、系统的通道以及系统的不可信数据项。其中系统的方法包含系统入口点和系统出口点，系统的入口点是系统从环境中接收输入数据的方法，系统的出口点是系统向其环境进行输出和发送数据的方法。在一般的软件系统中，操作系统即为其环境，软件系统向操作系统进行输出或数据发送一般需要进行系统调用，这些系统调用由操作系统提供给运行于操作系统上的软件系统使用。分别使用 M, C, I 表示一套软件系统 S 的方法、通道和不可信数据项，则该软件系统的攻击面可由以下三元组表示。

$$\text{surf}_S = M, C, I \quad (1)$$

将系统的方法部分进一步细分为系统入口点 M_{in} 和系统的出口点 M_{out} ，则其攻击面可由以下四元组表示。

$$\text{surf}_S = M_{\text{in}}, M_{\text{out}}, C, I \quad (2)$$

文献[19]根据采用特殊架构时系统攻击面会产生转移的特点提出了攻击面转移。文献[15]提出攻击面模型的构建是攻击面理论研究的重点,是准确评估系统安全性的关键,同时提出了一种对非相似冗余度信息系统进行建模的方法。非相似冗余度信息系统架构如图 2 所示。

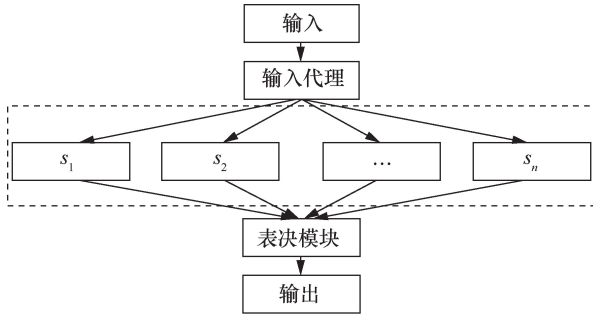


图 2 非相似冗余度信息系统架构

Figure 2 Architecture of dissimilar redundant information system

非相似冗余度信息系统架构包括输入模块,进行输入分发的输入代理模块,功能相同的异构执行体模块,进行结果表决的表决模块以及输出模块。假设有一非相似冗余度系统 S , 其由 $s_1, s_2, s_3, \dots, s_n$ 等异构执行体组成,若按照传统攻击面的建模方式进行建模,则系统攻击面 R_S 可由式(3)表示:

$$R_S = R_{s_1} \cup R_{s_2} \cup \dots \cup R_{s_n}, n \in N^* \quad (3)$$

其中, R_{s_i} 为异构执行体 s_i 的攻击面,由此可见采用非相似冗余度信息系统架构的攻击面反而变大,与实际情况不相符,因此文献[15]提出了一种加了表决机制的新型攻击面建模方式,将表决机制抽象为多个执行体攻击面取交集的过程。引入表决机制后系统 S 的攻击面可由式(4)表示。

$$R_S = R_{s_1} \cap R_{s_2} \cap \dots \cap R_{s_n}, n \in N^* \quad (4)$$

可以看出,改进后的模型在进行攻击面表决后,系统攻击面显著减小,与实际情况相符。然而,采用多变量进程架构的软件系统与传统的非相似冗余度架构 Web 应用系统相比,异构冗余粒度和表决粒度较小,并且采用多变量进程架构的软件系统主要针对系统出口点即系统调用进行表决,在多变量进程系统正常运行时,其攻击面并不会改变,但当遭受攻击时,触发表决会导致系统攻击面的动态变化。此过程无法通过简单取交

集的方式进行抽象,因此采用多变量进程架构的系统不能使用文献[15]中提出的非相似冗余度信息系统架构的建模方式。

2 攻击面模型

2.1 传统攻击面模型

对软件系统进行攻击面建模是度量软件系统安全性的重要步骤,文献[18]给出的传统攻击面的建模方式如下。

系统集 S_i 包含的一个系统 s_i 的环境 E_{s_i} 为一个三元组,如式(5)所示。

$$E_{s_i} = U_i, D_i, T_i \quad (5)$$

其中, U_i 为攻击者集合, T_i 为不含 s_i 的系统集合, D_i 为数据集集合,如图 3 所示。

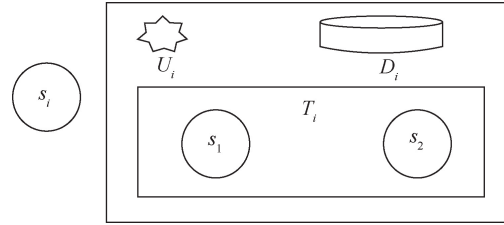


图 3 系统环境

Figure 3 System environment

则系统 s_i 的攻击面可由式(6)表示。

$$\text{surf}_{s_i} = M_{\text{in}}^{E_{s_i}}, M_{\text{out}}^{E_{s_i}}, C^{E_{s_i}}, I^{E_{s_i}} \quad (6)$$

定义系统 s_i 的攻击面资源集合 R_{s_i} 由式(7)表示。

$$R_{s_i} = M_{\text{in}}^{E_{s_i}} \cup M_{\text{out}}^{E_{s_i}} \cup C^{E_{s_i}} \cup I^{E_{s_i}} \quad (7)$$

2.2 多变量系统攻击面模型

本文对传统攻击面模型进行扩展,并构建多变量系统的攻击面模型;使用形式化方式表示多变量系统的攻击面,根据多变量系统在系统出口点处的表决机制对传统攻击面模型进行改进,使其能解释多变量系统攻击面缩小的现象。

在多变量系统中,多变量进程的异构性体现了异构冗余的思想,由于多个多变量进程之间采用了地址随机化技术,攻击者难以同时采用缓冲区溢出攻击对所有多变量进程进行控制流劫持,因此当攻击者通过控制流劫持某一进程多变量进行恶意系统调用时,其他未被攻击成功的多变量

进程仍维持正常的系统调用，表决模块拦截到各个多变量进程的系统调用并进行对比，从而阻止攻击。多变量系统中存在一致表决、近似表决等多种表决策略，本文的多变量系统表决策略采取一致表决，即所有多变量进程的系统调用一致时才表决通过，否则表决不通过。

根据多变量系统的特点，定义系统集 S 中的多变量进程功能相同且均能够独立完成系统功能，假设系统集 S 中存在 n 个等价的多变量进程，则记多变量进程 i 为 s_i ($s_i \in S, i \leq n$)。由于各个多变量进程独立存在，不存在相互通信，故可视之为多个独立的攻击目标，当系统集 S 受到攻击时，各个多变量进程同时成为被攻击的目标。若假设每个多变量进程的攻击面为 R_{s_i} ，则在进行表决之前，多变量系统的攻击面为 R_S 。根据传统攻击面理论，各个多变量进程的功能相同且共用系统资源，此时多变量系统的攻击面 R_S 相较于每一个多变量进程的攻击面 R_{s_i} 并没有显著变化，这与多变量系统能提高系统安全性的研究结论相悖。产生这种结果是因为传统攻击面理论中并没有针对多变量系统的架构引入系统出口表决机制。

传统攻击面理论中并未引入异构冗余的概念，而多变量系统安全性的提升主要是因为多变量系统采取了异构冗余的架构，该特性可以使多变量系统在系统出口点进行一致性表决。因此，本文在传统攻击面理论的基础上引入系统出口表决机制，相关定义如下。

定义 1 给定环境 $E = \langle U, D, T \rangle$ ，存在系统 x 和 y ，其攻击面分别为 R_x 和 R_y ，若系统 x 与 y 在方法、通道、数据 3 个方面完全一致，则称 x 与 y 同构，两者攻击面相同，记为 $|R_x| = |R_y|$ 。若系统 x 与系统 y 在方法、通道、数据 3 个方面的任何一方面不一致，则称 x 与 y 不同构，两者攻击面不相同，若 $R_x \subseteq R_y$ ，则称 $|R_y| > |R_x|$ ，系统 y 的攻击面比系统 x 的攻击面大。

定义 2 给定一个进程 s ，在其出口点存在对于操作系统方法的调用，即系统调用。设在操作系统中系统调用共有 n 种，则将操作系统调用抽象为 n 个两两垂直的单位向量 $\alpha_1 \cdots \alpha_n$ 。当两个系统调用不相同，代表其单位向量互相垂直，两

者点积结果为 0。在进程 s 执行过程中使用的系统调用向量按照顺序组成一个集合 T_s ，称为系统 S 的系统调用向量集合 T_S 。

定义 3 给定的一个多变量系统 S 中存在两个进程 s_1, s_2 ，其系统调用向量集合分别为 T_{s_1} 和 T_{s_2} ，集合中元素个数分别为 n_1 和 n_2 ，对 T_{s_1} 和 T_{s_2} 中对应位置的向量进行点积运算，记第一组运算结果不为零的向量之前的向量为集合 A ，如下所示：

$$A = \left\{ \beta | \beta \in \{\alpha_1 \cdots \alpha_k\}, \right. \\ \left. 1 \leq k \leq \min(n_1, n_2), \alpha_{k+1}^{T_{s_1}} \cdot \alpha_{k+1}^{T_{s_2}} = 0 \right\} \quad (8)$$

在多变量系统中，如果进程 s_1 和 s_2 均未受到攻击，则应该拥有相同的系统调用，即 T_{s_1} 与 T_{s_2} 相同。记两者不同的系统调用集合为 K ，如式(9)所示。

$$K = (T_{s_1} - A) \cup (T_{s_2} - A) \quad (9)$$

故该系统 S 出口点经过表决之后的攻击面集合如式(10)所示。

$$M_{\text{out表决后}}^S = M_{\text{out表决前}}^S - K \quad (10)$$

则系统 S 的攻击面描述如式(11)所示。

$$R_S = M_{\text{in}}^S, M_{\text{out表决前}}^S - K, C^S, I^S \quad (11)$$

可见经过系统出口点表决后多变量系统的攻击面缩小。

3 实例分析

本文采用两组多变量执行架构的软件系统进行实例分析，分别通过与未采用多变量架构的功能相同的软件系统进行攻击面的对比分析，体现多变量系统在攻击面上的变化。

实例采用的两种软件系统均使用了危险函数 `gets()`，攻击者能够通过缓冲区溢出漏洞对程序的控制流进行篡改，从而威胁系统的安全。

对正常运行时的多变量系统和普通系统进行建模比较，由于未受到攻击，多变量系统的各个进程在表决时系统调用序列相同，故此时多变量系统和普通系统的攻击面保持一致，如下所示：

$$K = (T_{s_1} - A) \cup (T_{s_2} - A) = \emptyset \quad (12)$$

$$M_{out表决后}^{S多变量} = M_{out表决前}^{S多变量} - \emptyset = M_{out表决前}^{S多变量} \quad (13)$$

$$R_{S多变量} = M_{in}^{S多变量}, M_{out表决前}^{S多变量}, \quad (14)$$

$$C^{S多变量}, I^{S多变量} = R_{S普通}$$

在正常运行时, 多变量系统和普通系统产生的系统调用如表 1 和表 2 所示。

表 1 第一组软件系统未受攻击的系统调用
Table 1 The first group of system call of software system not attacked

系统调用序号	普通系统	多变量系统 进程 1	多变量系统 进程 2
1	Brk	Brk	Brk
2	Mmap	Mmap	Mmap
3	Access	Access	Access
4	Open	Open	Open
5	Fstat	Fstat	Fstat
6	Mmap	Mmap	Mmap
7	Close	Close	Close
8	Open	Open	Open
9	Read	Read	Read
10	Fstat	Fstat	Fstat
11	Mmap	Mmap	Mmap
12	Mprotect	Mprotect	Mprotect
13	Write	Write	Write
14	Exit	Exit	Exit

可以看出, 当未受到攻击时, 多变量系统和普通软件系统的系统调用一致, 两者在系统出口点的攻击面相同, 这与建模分析结果一致。

当软件系统受到攻击时, 普通软件系统会遭到控制流劫持, 而多变量系统由于系统出口点表决的存在, 会中断软件的运行, 阻断攻击者的攻击行为。在这种情况下, 多变量系统会产生不同的系统调用, 导致表决生效, 使得采用多变量架构的软件系统攻击面变小, 对多变量系统和普通系统进行建模分析对比如下所示:

$$K = (T_{s_1} - A) \cup (T_{s_2} - A) \neq \emptyset \quad (15)$$

$$M_{out表决后}^{S多变量} = M_{out表决前}^{S多变量} - K < M_{out表决前}^{S多变量} \quad (16)$$

$$R_{S多变量} = M_{in}^{S多变量}, M_{out表决后}^{S多变量}, C^{S多变量}, I^{S多变量} < R_{S普通} \quad (17)$$

当软件系统受到攻击时, 两者的系统调用如表 3 和表 4 所示。

表 2 第二组软件系统未受攻击的系统调用
Table 2 The second group of system call of software system not attacked

系统调用序号	普通系统	多变量 系统进程 1	多变量 系统进程 2
1	Brk	Brk	Brk
2	Mmap	Mmap	Mmap
3	Access	Access	Access
4	Open	Open	Open
5	Fstat	Fstat	Fstat
6	Mmap	Mmap	Mmap
7	Close	Close	Close
8	Open	Open	Open
9	Fstat	Fstat	Fstat
10	Mmap	Mmap	Mmap
11	Close	Close	Close
12	Open	Open	Open
13	Read	Read	Read
14	Fstat	Fstat	Fstat
15	Mmap	Mmap	Mmap
16	Mprotect	Mprotect	Mprotect
17	Write	Write	Write
18	Exit	Exit	Exit

表 3 第一组软件系统受攻击的系统调用
Table 3 The first group of system call of software system attacked

系统调用序号	普通系统	多变量 系统进程 1	多变量 系统进程 2
1	Brk	Brk	Brk
2	Mmap	Mmap	Mmap
3	Access	Access	Access
4	Open	Open	Open
5	Fstat	Fstat	Fstat
6	Mmap	Mmap	Mmap
7	Close	Close	Close
8	Open	Open	Open
9	Read		
10	Write		
11	Mmap		
12	Write		
13	Write		
...	Fstat		

注: 表中的空白表示该时间进程已结束, 不再产生系统调用。

表4 第二组软件系统受攻击的系统调用
Table 4 The second group of system call of software system attacked

系统调用序号	普通系统	多变体 系统进程 1	多变体 系统进程 2
1	Brk	Brk	Brk
2	Mmap	Mmap	Mmap
3	Access	Access	Access
4	Open	Open	Open
5	Fstat	Fstat	Fstat
6	Mmap	Mmap	Mmap
7	Close	Close	Close
8	Open	Open	Open
9	Fstat	Fstat	Fstat
10	Mmap	Mmap	Mmap
11	Close	Close	Close
12	Open	Open	Open
13	Read		
14	Write		
15	Mmap		
16	Write		
17	Write		
...	Fstat		

分析两组实例可知，攻击者进行攻击后，普通软件系统由于没有系统出口点的表决，在 read 系统调用处被攻击，攻击者之后可以进行控制流劫持操作。而多变体系统在 read 系统调用处的表决结果不一致，从而强制中断程序的运行，避免了攻击者的劫持。多变体系统由于受到攻击，只执行了部分系统调用，系统的攻击面在遭受攻击时减小，这与前面的建模分析结果一致。

4 结束语

本文根据多变体系统的表决特点，在传统攻击面模型的基础上提出了多变体进程系统的攻击面模型，形式化地描述了多变体系统在遭受攻击时攻击面的动态变化过程。阐明了多变体系统在系统出口点的表决方式对攻击面变化的影响，解释了多变体系统受到攻击时系统攻击面动态变化的原理。通过两组实例分析，对比未遭受攻击时及遭受攻击时普通系统与多变体系统的攻击面，证明了对于多变体系统攻击面建模的合理性。

本文对多变体系统的攻击面建模进行了初步研究，理论部分仍不完善，未来的工作将集中于多变体系统攻击面的量化分析以及对采用不同表决方式的多变体系统的攻击面变化的分析。

参考文献：

- [1] 方滨兴. 定义网络空间安全[J]. 网络与信息安全学报, 2018, 4(1): 1-5.
FANG B X. Define cyberspace security[J]. Chinese Journal of Network and Information Security, 2018, 4(1): 1-5.
- [2] 邵思豪, 高庆, 马森, 等. 缓冲区溢出漏洞分析技术研究进展[J]. 软件学报, 2018, 29(5): 1179-1198.
SHAO S H, GAO Q, MA S, et al. Progress in research on buffer overflow vulnerability analysis technologies[J]. Journal of Software, 2018, 29(5): 1179-1198.
- [3] BLETSCH T, JIANG X X, FREEH V W, et al. Jump-oriented programming: a new class of code-reuse attack[C]//Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security - ASIACCS '11. 2011: 30-40.
- [4] LEVY E. Smashing the stack for fun and profit[J]. Phrack Maga-zine, 1996, 8(49): 1-25.
- [5] SHACHAM H. The geometry of innocent flesh on the bone: return-into-libc without function calls (on the x86)[C]//Proceedings of the 14th ACM conference on Computer and communications security. 2007: 552-561.
- [6] 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]. 网络与信息安全学报, 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.
- [7] COWAN C, PU C, MAIER D, et al. StackGuard: automatic adaptive detection and prevention of buffer-overflow attacks[C]//Proceedings of 7th USENIX Security Conference. 1998: 63-78.
- [8] COWAN C, BARRINGER M, BEATTIE S, et al. FormatGuard: automatic protection from printf format string vulnerabilities[C]//Proceedings of 10th USENIX Security Symposium. 2001: 13-17.
- [9] BITTAU A, BELAY A, MASHTIZADEH A, et al. Hacking blind[C]//Proceedings of 2014 IEEE Symposium on Security and Privacy. 2014: 227-242.
- [10] WANG Z, JIANG X X. HyperSafe: a lightweight approach to provide lifetime hypervisor control-flow integrity[C]//Proceedings of 2010 IEEE Symposium on Security and Privacy. 2010: 380-395.
- [11] 马博林, 张铮, 陈源, 等. 基于指令集随机化的抗代码注入攻击方法[J]. 信息安全学报, 2020, 5(4): 30-43.
MA B L, ZHANG Z, CHEN Y, et al. The defense method for code-injection attacks based on instruction set randomization[J]. Journal of Cyber Security, 2020, 5(4): 30-43.
- [12] COX B, EVANS D, FILIPI A, et al. N-variant systems a secretless framework for security through diversity[J]. 15th USENIX Security Symposium, 2006: 105-120.
- [13] SALAMAT B, JACKSON T, GAL A, et al. Orchestra: intrusion detection using parallel execution and monitoring of program variants in user-space[C]//Proceedings of the 4th ACM European conference on Computer systems. 2009: 33-46.

- [14] KONING K, BOS H, GIUFFRIDA C. Secure and efficient multi-variant execution using hardware-assisted process virtualization[C]//Proceedings of 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). 2016: 431-442.
- [15] 张铮, 王立群, 李卫超. 面向非相似冗余信息系统的攻击面模型[J]. 通信学报, 2018, 39(S2): 223-230.
ZHANG Z, WANG L Q, LI W C. Research on formal model for an information system's attack surface with dissimilar redundant architecture[J]. Journal on Communications, 2018, 39(S2): 223-230.
- [16] 潘传幸, 张铮, 马博林, 等. 面向进程控制流劫持攻击的拟态防御方法[J]. 通信学报, 2021, 42(1): 37-47.
PAN C X, ZHANG Z, MA B L, et al. Method against process control-flow hijacking based on mimic defense[J]. Journal on Communications, 2021, 42(1): 37-47.
- [17] 马博林, 张铮, 刘浩, 等. SQLMVED: 基于多变体执行的 SQL 注入运行时防御系统[J]. 通信学报, 2021, 42(4): 127-138.
MA B L, ZHANG Z, LIU H, et al. SQLMVED: SQL injection runtime prevention system based on multi-variant execution[J]. Journal on Communications, 2021, 42(4): 127-138.
- [18] MANADHATA P K, WING J M. A formal model for a system's attack surface[M]//Advances in Information Security. New York, NY: Springer New York, 2011: 1-28.
- [19] MANADHATA P K. Game theoretic approaches to attack surface shifting[M]//Moving Target Defense II. New York, NY: Springer New York, 2012: 1-13.

[作者简介]



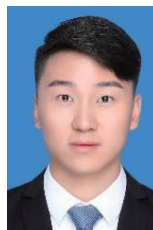
邢福康(1997-), 男, 山东聊城人, 信息工程大学博士生, 主要研究方向为网络空间安全、Web 应用安全。



张铮(1976-), 男, 湖北黄冈人, 博士, 信息工程大学副教授, 主要研究方向为网络空间安全、主动防御技术。



隋然(1974-), 男, 山东青岛人, 博士, 中央军委后勤保障部信息中心研究员, 主要研究方向为网络空间安全。



曲晟(1996-), 男, 山西忻州人, 信息工程大学博士生, 主要研究方向为网络空间安全、主动防御技术。



季新生(1968-), 男, 河南驻马店人, 博士, 信息工程大学教授、博士生导师, 主要研究方向为网络空间安全、无线通信。