计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.200100135

一种基于强化学习的嵌入式系统抗拒绝服务攻击的缓存调度方案

黄锦灏[1]　丁钰真[1]　肖　亮[1]　沈志荣[1]　朱珍民[2]
1 厦门大学信息学院　厦门 361005
2 中国科学院大学计算技术研究所　北京 100190
(506109624@qq.com)

摘　要　在多核嵌入式操作系统中,中央处理器对共享最后一级缓存(Last Level Cache,LLC)的资源调度决定了各用户进程的指令周期数(Instructions Per Cycle,IPC),以及对拒绝服务(Denial-of-Service,DoS)攻击的鲁棒性。但是,现有缓存调度方案依赖于具体的LLC调度模型和DoS攻击模型,使中央处理器难以在不同调度环境中的每个调度周期及时获得用户进程的运行信息。因此,文中提出一种基于强化学习的嵌入式系统LLC调度技术,以抵御拒绝服务攻击。该技术根据用户进程的LLC占用起始位置和终止位置,结合反馈的指令周期数、载入未命中率和存储未命中率等信息,优化LLC的占用位置和占用空间。在动态LLC调度环境下,中央处理器不需要预知DoS攻击模型,即可提高指令周期数并同时降低恶意进程的DoS攻击成功率。在多租户虚拟机共同参与的多核嵌入式操作系统中的仿真结果表明,所提技术可以显著提高指令周期数并降低DoS攻击的成功率。

关键词:缓存调度;DoS攻击;强化学习;嵌入式系统
中图法分类号　TP316

Reinforcement Learning Based Cache Scheduling Against Denial-of-Service Attacks in Embedded Systems

HUANG Jin-hao[1],DING Yu-zhen[1],XIAO Liang[1],SHEN Zhi-rong[1] and ZHU Zhen-min[2]
1 School of Informatics,Xiamen University,Xiamen 361005,China
2 Institute of Computing technology,Chinese Academy of Sciences University,Beijing 100190,China

Abstract　The sharing last level cache (LLC) scheduling of the central processor determines the instructions per cycle (IPC) of the user processes and the robustness of denial-of-service (DoS) attacks in the multicore embedded operating systems. However, existing scheduling schemes rely on the specific LLC scheduling model and DoS attack model, which makes it difficult for the processor to obtain the running information of the user processes in each scheduling cycle under different scheduling environments. Therefore, this paper proposes a reinforcement learning (RL) based LLC scheduling scheme to against DoS attacks in embedded systems, which optimizes the occupied position and the occupied space based on the measured occupied start and end positions, the previous IPC, load miss rate and store miss rate. The processor can jointly increase the IPC and reduce the success rate of the DoS attack from the malicious process without knowing the DoS attack model in the dynamic LLC scheduling environment. Simulations are implemented on the multicore embedded operating systems where multitenant virtual machines participate together, which show that the proposed scheme can significantly increase the IPC and reduce the success rate of the DoS attack.

Keywords　LLC scheduling,DoS attack,Reinforcement learning,Embedded systems

1　引言

缓存是介于中央处理器和内存之间的硬件单元,用于暂时存储中央处理器的运行数据。中央处理器向内存单元发出访问请求时会先查看缓存单元中是否存在该请求数据,若存在则表示命中且不需要再访问内存单元[1]。相较于直接访问内存,中央处理器先到缓存中查看是否有请求数据,可以有效减少数据读写操作的平均时间并节省自身能耗[2]。随着多核操作系统和多租户技术的出现,不同用户可以在同一台服务器上共享以路(way)为基本单位划分的最后一级缓存资源[3-4]。各用户进程可以占用其中某几个连续的way,从而提高中央处理器对LLC的利用率[5]和中央处理器自身的指令周期数等计算性能,但各个用户进程之间的资源竞争会导致性能干扰[6]。因此,可以合理分配资源从而减小各用户共享

到稿日期:2020-01-21　返修日期:2020-05-21　本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:国家自然科学基金(61971366,61671396)
This work was supported by the National Natural Science Foundation of China (61971366,61671396).
通信作者:肖亮(lxiao@xmu.edu.cn)

资源时产生的干扰的缓存调度技术成为研究热点[7]。另外,在中央处理器调度资源和分配缓存的过程中,可能存在一些恶意进程对用户进程实施缓存攻击,故意占用大量缓存空间,从而影响用户进程的工作效率,甚至导致拒绝服务发生而使用户进程无法正常工作[8-9]。因此,如何在中央处理器的LLC调度优化过程中提高用户进程的指令周期数并降低恶意进程的DoS攻击成功率,成为缓存调度技术发展的关键问题之一。

现今有很多关于缓存调度技术的研究工作。例如,中央处理器根据用户进程LLC的占用大小来预测未命中率,结合预测结果并利用模拟退火技术在实验平台为各用户进程提供LLC调度方案[10];中央处理器实时监测用户进程对LLC资源的需求,在保证所有用户进程基本性能的基础上动态调整LLC调度方案[11]。针对当前多核处理器在共享计算资源时容易受到DoS攻击的情况[12],中央处理器可以基于已知的攻击模型优化内存带宽等参数[9],或者定期调整进程的优先级,尽可能实现用户进程缓存资源的公平分配[13],也可采用新型缓存替代策略减少攻击者占用的缓存空间[14],从而避免出现拒绝服务。然而,中央处理器在动态的运行环境中难以精确预估用户进程的计算性能以及攻击者的信息,因此难以据此来优化用户进程的执行效率和抗DoS攻击的能力。强化学习技术,如Q学习[15]、深度Q网络[16]等,可以应用在资源调度和安全防御的动态博弈过程中以优化调度策略和安全抵御策略。例如,在资源调度方面,中央处理器在未知系统空闲周期模型时利用Q学习来动态优化电源模式选择,从而对电源能耗和运行时间进行权衡[17];多核处理器系统利用多智能体Q学习动态优化多级缓存分配策略,从而提高系统的吞吐量并缩短运行时延[18]。在安全方面,物联网医疗设备利用Hotbooting Q学习动态优化传感数据的上传策略,从而提高用户数据的隐私水平[19];中央处理器利用深度Q网络在未知攻击模型和数据存储模型的情况下动态优化资源分配策略,从而增强云存储数据的保护水平[20]。

由于当前缓存调度策略的选取仅与当前时刻的系统状态有关而与历史状态无关,中央处理器对运行的各个目标进程进行LLC调度的过程可以建模为马尔可夫决策过程,故强化学习技术可以用来选择最优的缓存分配参数。因此,本文提出了一种基于强化学习的嵌入式系统抗拒绝服务攻击的缓存调度方案。该方案可使中央处理器根据用户进程反馈的指令周期数、载入未命中率和存储未命中率等信息,利用强化学习算法在不需要预知缓存分配模型和攻击模型的情况下,通过不断试错来动态优化用户进程的LLC占用位置和占用空间,在提高用户进程指令周期数的同时降低攻击者的DoS攻击成功率,从而提高中央处理器的效益。

2 抗拒绝服务攻击的缓存调度技术模型

本文考虑一个由多租户虚拟机共同参与的多核嵌入式系统。该嵌入式系统可以用来完成各种不同的特定任务,如文件压缩任务和系统并行计算任务。数据中心是嵌入式系统中的一个多核共享资源平台,可供不同租户虚拟机共享网络和LLC等资源。中央处理器首先通过指令将不同租户虚拟机运行所产生的不同进程与各个中央处理器的物理核绑定在一起,以实现进程间的隔离,然后为各个目标进程分配不同的LLC资源,最后根据各个目标进程反馈的信息来计算指令周期数、载入未命中率和存储未命中率,并在这个过程中应用性能监视器实时监测并获取各个目标进程的需求。

2.1 缓存分配模型

如图1所示,中央处理器在硬件层面实现各目标进程与特定物理核的绑定操作,使每个中央处理器的物理核只与独立的一段连续LLC相关联。在 k 时刻,中央处理器为 N 个物理核分配LLC单元,使该物理核上运行的目标进程只能在与之关联的LLC单元上进行数据读写。为了实现在软硬件层面上将LLC单元、物理核以及运行的目标进程同时关联,中央处理器将物理核以及运行的目标进程当成一个整体并划分为不同的服务等级,应用性能监视器提取不同服务等级中各个进程在一段时间内的运行数据,包括目标进程 $i(1\leqslant i\leqslant N)$ 执行的指令数 $n_i^{(k)}$ 以及运行的周期数 $c_i^{(k)}$、在LLC单元的载入未命中数 $m_i^{(k)}$ 以及LLC的总载入数 $l_i^{(k)}$、在LLC单元的存储未命中数 $t_i^{(k)}$ 以及LLC的总存储数 $s_i^{(k)}$,并通过计算获得所有目标进程的指令周期数 $\eta^{(k)}=\sum_{i=1}^{N}n_i^{(k)}/c_i^{(k)}$、载入未命中率 $\lambda^{(k)}=\sum_{i=1}^{N}m_i^{(k)}/l_i^{(k)}$ 和存储未命中率 $\gamma^{(k)}=\sum_{i=1}^{N}t_i^{(k)}/s_i^{(k)}$,从而对目标进程的运行性能进行实时监控。

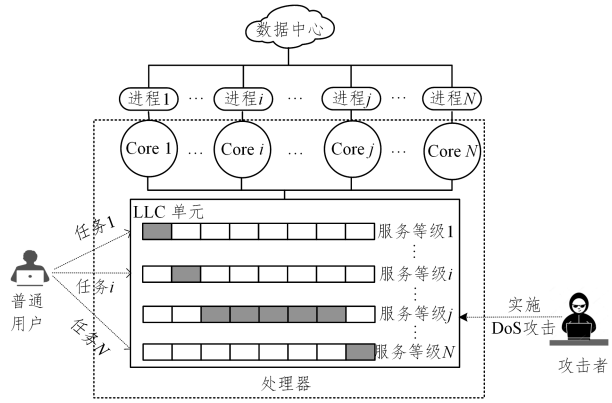


图1 抗拒绝服务攻击的缓存调度技术模型

Fig. 1 Model of cache scheduling scheme against DoS attacks

2.2 攻防模型

中央处理器采用完全独立的缓存分配方案,在保证各目标进程占用的LLC资源相对独立的情况下将LLC分配给所有的目标进程。将目标进程 i 在LLC单元被分配的起始位置记为 $\varphi_i^{(k)}$,终止位置记为 $\varphi_i^{(k)}$,则占用空间为 $o_i^{(k)}=\varphi_i^{(k)}-\varphi_i^{(k)}$。考虑到在分配的过程中存在某个恶意进程 j 实施DoS攻击,占用大量LLC资源,导致目标进程占用LLC资源的数量减少,处理任务的效率降低,甚至出现部分目标进程无法分配到缓存资源,从而无法提供正常服务的情况。定义恶意进程实施DoS攻击的成功率为 $P_e^{(k)}=\mathrm{I}(o_i^{(k)}\neq 0,\forall i\in[1,N])$,其中 $\mathrm{I}(\cdot)$ 为指示函数,括号内条件成立时取0,否则取1。当

$o_i^{(k)}$ 为 0 时表示中央处理器分配的大多数 LLC 资源被恶意进程侵占，导致目标进程 i 无法占用资源，从而产生拒绝服务的情况。

中央处理器根据目标进程的指令周期数、载入未命中率和存储未命中率等运行信息以及当前时刻在 LLC 上的占用位置，为每个目标进程分配合理大小的缓存空间，以降低目标进程和潜在恶意进程共同使用相同缓存的概率，从而降低恶意进程对目标进程发起 DoS 攻击的成功率。

3 基于强化学习的嵌入式系统抗拒绝服务攻击的缓存调度方案

本文提出一种基于强化学习的嵌入式系统抗拒绝服务攻击的缓存调度(Reinforcement Learning Based Cache Scheduling Against DoS Attacksin Embedded Systems, RLCSD)方案。如算法 1 所示，该方案动态选择目标进程 i 在 LLC 单元的占用位置 $\boldsymbol{x}_i^{(k)}=[\phi_i^{(k)},\varphi_i^{(k)}]$ 并确定占用空间 $o_i^{(k)}$，从而提高目标进程运行的指令周期数并降低恶意进程实施 DoS 攻击的成功率。具体地，中央处理器根据目标进程的反馈信息计算上一时刻的指令周期数 $\eta^{(k-1)}$、载入未命中率 $\lambda^{(k-1)}$ 和存储未命中率 $\gamma^{(k-1)}$，观测上一时刻目标进程的占用位置 $\phi_i^{(k-1)}$ 和 $\varphi_i^{(k-1)}$，将缓存分配的状态构建为 $\boldsymbol{s}^{(k)}=[\eta^{(k-1)},\lambda^{(k-1)},\gamma^{(k-1)},\phi^{(k-1)},\varphi^{(k-1)}]$。中央处理器根据当前时刻的系统状态采用 ε 贪婪策略动态选择 LLC 分配策略 $\boldsymbol{x}^{(k)}=[\boldsymbol{x}_i^{(k)}]_{1\leqslant i\leqslant N}$，即以 $1-\varepsilon$ 的大概率选择最大化 Q 函数 $Q(\boldsymbol{s}^{(k)},\boldsymbol{x}^{(k)})$ 的 LLC 分配策略 $\boldsymbol{x}^{(k)}$，并保留极小的概率 ε 进行探索，即随机选择其他可能的 LLC 分配策略，其中 $0<\varepsilon<1$，此举可用来平衡学习过程中的探索和利用。Q 函数 $Q(\boldsymbol{s}^{(k)},\boldsymbol{x}^{(k)})$ 在此是用来描述当前状态 $\boldsymbol{s}^{(k)}$ 下选择动作 $\boldsymbol{x}^{(k)}$ 的长期累积折扣奖赏，由全连接层组成的深度神经网络拟合获得。深度神经网络的输入 $\psi^{(k)}$ 由以往的 V 个状态动作对和当前时刻的系统状态 $\boldsymbol{s}^{(k)}$ 组成，即 $\psi^{(k)}=\{\boldsymbol{s}^{(k-V)},\boldsymbol{x}^{(k-V)},\cdots,\boldsymbol{x}^{(k-1)},\boldsymbol{s}^{(k)}\}$。

在中央处理器选取缓存分配策略后，各目标进程在分配的资源上运行以完成不同任务，并将运行的指令数、周期数等相关信息反馈给中央处理器。中央处理器根据各目标进程反馈的运行信息计算得到指令周期数 $\eta^{(k)}$、载入未命中率 $\lambda^{(k)}$ 和存储未命中率 $\gamma^{(k)}$，并通过检测目标进程是否成功反馈运行信息来测量 DoS 攻击的成功率 $P_e^{(k)}$，然后用如下公式评估效益 $u^{(k)}$：

$$u^{(k)}=\eta^{(k)}-\sigma\lambda^{(k)}-\tau\gamma^{(k)}-\upsilon P_e^{(k)} \tag{1}$$

其中，对系统各性能进行权衡的参数 σ 为载入未命中率系数，τ 为存储未命中率系数，υ 为 DoS 攻击成功率系数。从式(1)可以看出，中央处理器的效益与指令周期数呈正相关，而与载入未命中率和存储未命中率呈负相关。这是因为中央处理器在一个周期中运行的指令数越多效率就越高，而未命中则说明无法从缓存中获取想要的数据，从而需要访问内存，使得时延和能耗增大。中央处理器的效益与恶意进程的 DoS 攻击成功率呈负相关。这是因为目标进程受到 DoS 攻击后不能完成目标任务，轻则影响用户体验，重则带来严重后果。

该方案还利用经验回放技术，将经验 $\boldsymbol{e}^{(k)}=(\boldsymbol{s}^{(k)},\boldsymbol{x}^{(k)},u^{(k)},\boldsymbol{s}^{(k+1)})$ 放入经验池 D 中，每次从中随机抽取 B 个样本，计算损失函数 $L(\theta)$，并利用随机梯度下降的方法来更新 Q 网络的权重参数 θ。损失函数如下所示：

$$L(\theta)=E_{\boldsymbol{e}^{(d)}\in B}[(u^{(d)}+\delta\max_{x'}Q(\boldsymbol{s}^{(d+1)},\boldsymbol{x}')-Q(\boldsymbol{s}^{(d)},\boldsymbol{x}^{(d)}))^2] \tag{1}$$

其中，$\delta\in(0,1]$ 为折扣因子，表示对未来效益的不确定性，δ 越大则表明中央处理器越重视未来效益。

算法 1 基于强化学习的嵌入式系统抗拒绝服务攻击的缓存调度(RLCSD)算法

1. 初始化 θ，δ，$D=\varnothing$，$B=16$，$\phi^{(0)}$ 和 $\varphi^{(0)}$
2. 初始化效益中的权衡参数 σ，τ 和 υ
3. For k=1,2,… do
   3.1. 接收用户进程反馈信息并计算 $\eta^{(k-1)}$，$\lambda^{(k-1)}$ 和 $\gamma^{(k-1)}$；
   3.2. 观测用户进程的起始位置和终止位置 $\phi^{(k-1)}$，$\varphi^{(k-1)}$；
   3.3. 构建当前时刻的系统状态 $s^{(k)}=[\eta^{(k-1)},\lambda^{(k-1)},\gamma^{(k-1)},\phi^{(k-1)},\varphi^{(k-1)}]$；
   3.4 根据 ε 贪婪策略选取 LLC 调度策略 $x^{(k)}=[x_i^{(k)}]_{1\leqslant i\leqslant N}$；
   3.5. 各用户进程根据分配的占用位置和空间来运行完成任务；
   3.6. 获取运行后用户进程的信息得到 $\eta^{(k)}$，$\lambda^{(k)}$ 和 $\gamma^{(k)}$，并测量 $P_e^{(k)}$；
   3.7. 根据式(1)评估中央处理器的效益 $u^{(k)}$；
   3.8. 将经验 $\boldsymbol{e}^{(k)}=(\boldsymbol{s}^{(k)},\boldsymbol{x}^{(k)},u^{(k)},s^{(k+1)})$ 放入经验池 D 中；
   3.9. 从经验池 D 中随机选取 B 个经验；
   3.10. 根据式(2)更新 Q 网络的参数 θ；
4. End for

4 仿真分析

本节在 Pycharm 集成开发环境中使用 Python 语言对提出的 RLCSD 方案进行仿真分析，以 ubuntu 16.04 嵌入式操作系统为数据采集的实验环境。如图 2 所示，LLC 单元中 way 的个数设置为 8。数据中心运行的用户进程采用与文献[10]一样的标准性能评估公司(Standard Performance Evaluation Corporation, SPEC)的 CPU 子系统评估软件 SPEC CPU 2006 的测试项目，具体包括压缩文件应用 401.bzip2、人工智能围棋应用 445.gobmk、分子系统并行计算应用 444.namd 以及量子化学计算应用 416.gamess。中央处理器分别将其与 4 个物理核 Core 1－Core 4 进行关联，并将其划分为 4 个服务等级。

仿真采用完全独立的缓存调度方案，其数据来源是 ubuntu 系统中各目标进程在不同 way 上的实测数据。具体地，本文首先在 CPU 型号为 E5-2680 v4 的 Dell T630 服务器上对上述 4 个测试项目进行实验，获得不同 way 分配数对应的指令周期数、载入未命中率和存储未命中率等运行数据，然后基于实测运行数据在仿真平台进行仿真分析。仿真中，采用 kaiming 均匀分布对 Q 网络权重参数进行初始化[21]，将折扣因子 δ 设置为 0.01，并在 way 取值为 1～8 时随机初始化目标进程在 LLC 单元的起始位置 $\phi^{(0)}$ 以及终止位置 $\varphi^{(0)}$。

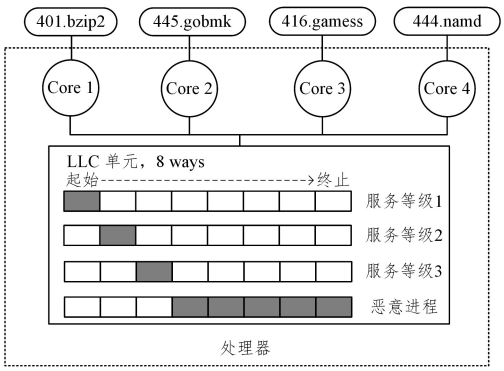
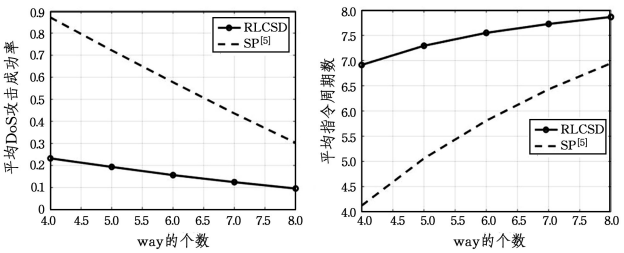


图 2 仿真设置图

Fig. 2 Simulation setting diagram

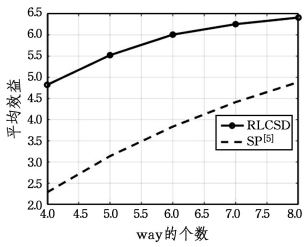
仿真中,设置恶意进程为伪装成测试项目 444.namd 的正常用户进程,和其他用户进程共享 LLC 资源并通过抢夺 LLC 资源实施 DoS 攻击,从而破坏其他用户进程的正常运行。本方案与文献[14]一样,不需检测潜在的恶意进程,只是通过对用户进程的指令周期数、载入未命中率和存储未命中率等运行信息和占用 LLC 位置的行为进行实时检测,动态调整各用户进程占用 LLC 的情况,在保证各用户进程都能提供正常服务的基础上实现整个系统效益的最大化。

图 3 展示了本文所提方案和文献[5]提出的固定参数(Static Parameter,SP)静态缓存分配方案的性能对比结果。具体地,当 way 的个数在 4~8 之间变化时,观察恶意进程的 DoS 攻击成功率、目标进程的指令周期数和中央处理器的效益这 3 个评估指标在 6000 个时隙上取平均值的结果。



(a)平均 DoS 攻击成功率

(b)目标进程的平均指令周期数



(c)中央处理器的平均效益

图 3 中央处理器 LLC 调度仿真结果图

Fig. 3 Simulation results of LLC scheduling diagram

如图 3(a)所示,在 way 的个数从 4 增加到 8 时,本文提出的方案能够使恶意进程的 DoS 攻击成功率下降 58.9%。这是因为增大 way 的个数能增加目标进程在 LLC 资源上的分配空间,降低目标进程受到恶意进程攻击从而产生拒绝服务的概率。当 way 的个数为 7 时,本文所提方案恶意进程的 DoS 攻击成功率为 0.1242,SP 方案恶意进程的 DoS 攻击成功率为 0.4359,与 SP 方案相比本文方案的 DoS 攻击成功率下降了71.5%。当 way 的个数为 4 时,本文所提方案和 SP 方案的 DoS 攻击成功率分别为 0.23 和 0.87,即本文方案在 way 的个数较少时仍可以保持较低的 DoS 攻击成功率,对 DoS 攻击有更好的鲁棒性。如图 3(b)所示,当 way 个数为 4 时,本文方案指令周期数为 6.9123,way 个数为 8 时指令周期数为 7.8635,所以相对于 way 个数为 4 时,way 个数为 8 能使指令周期数上升 13.8%。这是因为 way 的个数增大意味着目标进程存放临时数据的缓存空间更大,从而中央处理器在进行数据读写等操作时可以减小对内存的访问频率,加快目标进程运行指令的速度。当 way 个数为 7 时,本文提出的方案的指令周期数为 7.7249,SP 方案的指令周期数为 6.4298,与 SP 方案相比本文提出的方案的指令周期数提高了 20.1%。如图 3(c)所示,当 way 个数为 4 时,本文方案的效益为 4.8215,way 个数为 8 时效益为 6.4094,相比 way 个数为 4 时,way 个数为 8 能使效益上升 32.9%。这是因为与效益呈正相关的指令周期数有所提高,与效益呈负相关的未命中率和 DoS 攻击成功率有所下降,因此中央处理器的效益得到优化。当 way 个数为 7 时,本文方案的效益为 6.2535,SP 方案效益为 4.4.132,与 SP 方案相比本文方案能使效益提升 41.7%。

综上所述,本文提出的 RLCSD 方案的性能明显优于文献[5]中基于 SP 的静态缓存分配方案,即能够更有效地增加目标进程的指令周期数并提高中央处理器的效益,更显著地降低恶意进程的 DoS 攻击成功率。

结束语 针对嵌入式系统中的资源调度和抗拒绝服务攻击的问题,本文提出一种基于强化学习的抗拒绝服务攻击的缓存调度技术,在未知 LLC 调度模型和 DoS 攻击模型的情况下,动态优化目标进程在 LLC 上的占用起始位置和终止位置,从而提高目标进程的运行性能以及对 DoS 攻击的鲁棒性。仿真结果表明,当多核嵌入式操作系统中 LLC 单元有 4~8 个 way 时,本文所提 RLCSD 方案能使恶意进程的 DoS 攻击成功率下降 58.9%,并使目标进程的指令周期数提高 13.8%。当多核嵌入式操作系统中的 LLC 单元有 7 个 way 时,与文献[5]中的 SP 方案相比,RLCSD 方案能使恶意进程的 DoS 攻击成功率下降 71.5%,并使目标进程的指令周期数提高 20.2%。未来研究的重点是对缓存进行更细粒度的划分,并将所提算法应用于真实环境,通过实验验证其对用户进程计算性能的提升以及对 DoS 攻击的抵御效果。

参 考 文 献

[1] DETTI A,BRACCIALE L,LORETI P,et al. Modeling LRU cache with invalidation [J]. Computer Networks,2018,134(7):55-65.

[2] UDIPI A N,MURALIMANOHAR N,BALSUBRAMONIAN R,et al. Combining memory and a controller with photonics through 3D-stacking to enable scalable and energy-efficient systems[C]// International Symposium on Computer Architecture. ACM,2011:425-436.

[3] BROCK J, YE C, DING C, et al. Optimal cache partition-sharing [C]// International Conference on Parallel Processing. IEEE, 2015: 749-758.

[4] CHANG J, SOHI G S. Cooperative cache partitioning for chip multiprocessors[C]// International Conference on Supercomputing. ACM, 2014: 402-412.

[5] HERDRICH A, VERPLANKE E, AUTEE P, et al. Cache QoS: From concept to reality in the Intel® Xeon® processor E5-2600 v3 product family[C]// International Symposium on High Performance Computer Architecture. IEEE, 2016: 657-668.

[6] OTOOM M, JALEEL A, TRANCOSO P. Using personality metrics to improve cache interference management in multicore processors[C]// ACM International Conference on Computing Frontiers. ACM, 2017: 251-254.

[7] KASTURE H, SANCHEZ D. Ubik: efficient cache sharing with strict qos for latency-critical workloads[C]// Architectural Support for Programming Languages and Operating Systems. ACM, 2014: 729-742.

[8] GRUSS D, MAURICE C, WAGNER K, et al. Flush+ Flush: a fast and stealthy cache attack[C]// Detection of Intrusions and Malware, and Vulnerability Assessment. Berlin: Springer, 2016: 279-299.

[9] BECHTEL M, YUN H. Denial-of-service attacks on shared cache in multicore: Analysis and prevention[C]// Real-Time and Embedded Technology and Applications Symposium. IEEE, 2019: 357-367.

[10] XIANG Y, WANG X, HUANG Z, et al. DCAPS: dynamic cache allocation with partial sharing[C]// EuroSys. ACM, 2018: 13.

[11] XU C, RAJAMANI K, FERREIRA A, et al. dCat: dynamic cache management for efficient, performance-sensitive infrastructure-as-a-service[C]// EuroSys. ACM, 2018: 14.

[12] MUTLU T M O. Memory performance attacks: Denial of memory service in multi-core systems[C]// USENIX Security Symposium. USENIX, 2007: 18.

[13] KIM Y, PAPAMICHAEL M, MUTLU O, et al. Thread cluster memory scheduling: Exploiting differences in memory access behavior[C]// IEEE/ACM International Symposium on Microarchitecture. ACM, 2010: 65-76.

[14] KERAMIDAS G, PETOUMENOS P, KAXIRAS S, et al. Preventing denial-of-service attacks in shared CMP caches[C]// International Workshop on Embedded Computer Systems. Berlin: Springer, 2006: 359-372.

[15] WATKINS C J C H, DAYAN P. Q-learning [J]. Machine Learning, 1992, 8(3-4): 279-292.

[16] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Human-level control through deep reinforcement learning [J]. Nature, 2015, 518(7540): 529.

[17] YE R, XU Q. Learning-based power management for multicore processors via idle period manipulation [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2014, 33(7): 1043-1055.

[18] JAIN R, PANDA P R, SUBRAMONEY S. A coordinated multi-agent reinforcement learning approach to multi-level cache co-partitioning[C]// Design, Automation & Test in Europe Conference & Exhibition. IEEE, 2017: 800-805.

[19] MIN M, WAN X, XIAO L, et al. Learning-based privacy-aware offloading for healthcare IoT with energy harvesting [J]. IEEE Internet of Things Journal, 2018, 6(3): 4307-4316.

[20] MIN M, XIAO L, XIE C, et al. Defense against advanced persistent threats in dynamic cloud storage: A colonel blotto game approach [J]. IEEE Internet of Things Journal, 2018, 5(6): 4250-4261.

[21] HE K, ZHANG X, REN S, et al. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification [C]// IEEE International Conference on Computer Vision. IEEE, 2015: 1026-1034.

HUANG Jin-hao, born in 1996, postgraduate. His main research interests include network security and wireless communication.

XIAO Liang, born in 1980, Ph.D, professor, Ph. D supervisor, is a member of China Computer Federation. Her main research interests include network security and wireless communication.