CN 43-1258/TP
ISSN 1007-130X

计算机工程与科学
Computer Engineering & Science

第 46 卷第 3 期 2024 年 3 月
Vol. 46, No. 3, Mar. 2024

文章编号:1007-130X(2024)03-0440-13

结合决策树和 AdaBoost 的缓存侧信道攻击检测*

李　扬[1,2],尹大鹏[1],马自强[1,2],姚梓豪[1,2],魏良根[1,2]

(1.宁夏大学信息工程学院,宁夏 银川 750021;2.宁夏大数据与人工智能省部共建协同创新中心,宁夏 银川 750021)

摘　要:缓存侧信道攻击严重威胁各类系统的安全,对攻击进行检测可以有效阻断攻击。为此,提出了一种基于决策树和 AdaBoost 的 AD 检测模型,通过匹配系统硬件事件信息特征,快速有效地识别缓存侧信道攻击。首先,分析缓存侧信道攻击特点,提取攻击硬件事件特征模式。其次,利用决策树的快速响应能力,同时结合 AdaBoost 对数据样本进行加权迭代,对采集的不同负载下的特征数据进行模型训练,优化检测模型在不同负载时的整体检测精度。实验结果表明,该模型在不同系统负载条件下的检测精度均不低于 98.8%,能够快速准确地检测出缓存侧信道攻击。

关键词:系统安全;缓存侧信道攻击;机器学习;检测方法

中图分类号:TP393.08　　**文献标志码**:A

doi:10.3969/j.issn.1007-130X.2024.03.006

Cache side-channel attack detection combining decision tree and AdaBoost

LI Yang[1,2], YIN Da-peng[1], MA Zi-qiang[1,2], YAO Zi-hao[1,2], WEI Liang-gen[1,2]

(1. School of Information Engineering, Ningxia University, Yinchuan 750021;
2. Collaborative Innovation Center for Ningxia Big Data and Artificial Intelligence Co-founded by Ningxia Municipality and Ministry of Education, Yinchuan 750021, China)

Abstract: Cache side-channel attacks pose a serious threat to the security of various systems, and detecting the attacks can effectively block the attacks. Therefore, an AD detection model based on decision tree and AdaBoost is proposed to quickly and effectively identify cache side-channel attacks by matching system hardware event information features. Firstly, the characteristics of cache side-channel attacks are analyzed, and attack hardware event feature patterns are extracted. Secondly, the decision tree's rapid response capability is utilized, combined with AdaBoost's weighted iterative learning of data samples, to train the model on different load conditions. The model is optimized to improve the overall detection accuracy under different loads. Experimental results show that the detection accuracy of this model under different system load conditions is not less than 98.8%, and it can quickly and accurately detect cache side-channel attacks.

Key words: system security; cache side-channel attack; machine learning; detection method

1　引言

侧信道技术是一种利用密码设备的物理信息来窃取密钥的方法。自 1996 年 Kocher 等[1]首先提出侧信道攻击以来,该技术已经从需要外部设备辅助完成的接触式本地攻击[2-4],发展到以缓存侧信道攻击为代表的,通过攻击进程获取中央处理器、高速缓存等组件敏感信息的非接触式远程攻击[5-18]。处理器和高速缓存分别作为数据的处理

* **收稿日期**:2023-07-14;**修回日期**:2023-09-12
基金项目:宁夏回族自治区重点研发计划(2021BEB04047,2022BDE03008);宁夏自然科学基金(2021AAC030781)
通信作者:马自强(maziqiang@nxu.edu.cn)
通信地址:750021 宁夏银川市宁夏大学信息工程学院
Address: School of Information Engineering, Ningxia University, Yinchuan 750021, Ningxia, P. R. China

核心和常用数据存储核心,不可避免地包含系统与用户的各项敏感信息。通过缓存侧信道,恶意程序能使敏感信息跨进程[9-11]、跨虚拟机泄漏[8,10],甚至破坏可信执行环境[12,13],且具有不易发现、极难防御的特点。鉴于上述情况,针对缓存侧信道攻击的防御技术应运而生。攻击检测作为防御技术的重要组成部分,经过近二十年的发展,已有许多研究成果[19-33]。

早期常用的是基于签名或异常的检测方法[19-23],但是其静态或动态分析会带来较大性能开销,且易受到恶意软件的反分析或逃逸技术干扰,导致检测精度低。后来,使用硬件性能计数器 HPCs(Hardware Performance Counters)的检测方法[24],通过获取 HPCs 记录的缓存事件信息判断攻击是否发生,但该方法需要人工分析和选择特征,攻击变化时还需重新挑选特征。

随着人工智能技术,如机器学习在各领域广泛应用,研究人员开始将其引入缓存侧信道攻击的检测中,并提出了多种相关方法[25-32]。然而,大多数方法在实时性和应用场景方面考虑得较少[25,28,29,31]。缓存侧信道攻击可以在较短时间内完成,因此为避免敏感信息泄露,需及时检测出攻击,这对模型的检测速度和实时性提出了较高要求。同时,还需充分考虑处理器差异和负载变化等环境部署因素。部分机器学习检测技术在高负载情况下会出现较低检测精度[34,35]和较高误报率[36,37],甚至导致系统整体性能下降[38-41]。因此,在不同负载环境下实现高准确率和低系统消耗的攻击检测显得尤为重要。

为解决上述问题,本文以决策树和 AdaBoost 为基础提出一种结合决策树和 AdaBoost 的检测模型,用于检测缓存侧信道攻击中具有代表性的 2 类攻击:Flush + Reload[16] 和 Prime + Probe[17]。这 2 类攻击分别利用缓存争用和数据重用产生信息泄漏。

本文的主要工作包括以下 3 个方面:

(1)提出了一种结合决策树和 AdaBoost 的组合模型 DA(Decision tree-AdaBoost),用于缓存侧信道攻击检测。该模型考虑了被检测系统的特点,能够满足多负载情况下的检测需求,解决了单决策树模型在高负载下出现的数据重叠和检测精度下降等问题。

(2)最小化监测事件集。本文从冗杂的硬件事件信息中选取关键特征进行监测,避免了因监测大量特征而造成的性能下降。

(3)构建了适应现实攻击场景的数据集。通过监测硬件性能计数器的事件信息,针对不同负载情况,构建相应数据集用于模型训练和检测,保证了模型在不同负载下具备高精度、低开销的攻击检测能力。

本文剩余部分的组织结构如下:第 2 节介绍检测方法的整体框架;第 3 节阐述数据集的构建方法和特征选择策略;第 4 节介绍 DA 模型的设计思路和构建方法;第 5 节评估 DA 模型在不同负载情况下的检测性能;第 6 节总结全文。

2　检测方法的整体框架

本节介绍 DA 检测模型的整体架构。如图 1 所示,该模型由 3 部分组成:数据采集模块、数据集构建模块和攻击检测模块。

数据采集模块利用系统调用 *perf_event_open* 读取并统计 HPCs 中的事件信息;之后,将其输入数据集构建模块进行数据处理;最后,将处理完成的数据输入检测模块进行分类,若分类结果中有攻击类数据,则判断系统受到了缓存侧信道攻击。

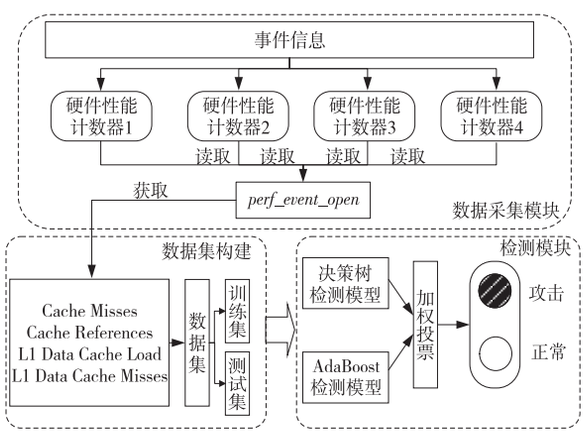

Figure 1　Overall framework of DA detection model

图 1　DA 检测模型整体框架

具体工作流程如下:

首先,通过 *perf_event_open* 获取硬件事件数据,并从中剔除无关或相关性较小的特征,最后筛选出与攻击相关性较大的重要特征用于模型训练。

筛选出重要特征后,数据采集模块再次使用 *perf_event_open* 在 HPCs 上重点筛选出重要特征。同时,分别统计在无系统负载和系统满载时攻击与无攻击的数据样本,并将这些数据样本传递给数据集构建模块。该模块将整合和预处理采集到的重要特征信息,形成最终的数据集。

最后,将构建完成的数据集输入检测模块。该模块使用决策树检测模块和 AdaBoost 检测模块

进行统一的训练和预测。同时,根据系统负载情况实时调整2个模型的权重,最终输出整体分类结果。若检测出攻击,则及时报告安全人员并启动相应缓解措施。

3 检测数据集构建

DA模型将HPCs事件作为检测缓存侧信道攻击的特征。然而,现代计算机中HPCs数量通常达几百个,若监测和统计事件信息过多,会增加系统负载,进而影响系统性能。因此,需要从众多硬件事件中选取能代表Flush+Reload和Prime+Probe攻击的特征。本节将介绍如何通过HPCs采集数据,并筛选出有利于识别攻击的重要特征。

3.1 数据样本获取

调用 *perf_event_open* 运行算法1,采集HPCs的硬件事件。采样过程中,*perf_event_open* 每次统计一个周期内所监测事件发生的次数。为了准确获得攻击时各事件的数据模式,本文在攻击发生的时间段内完成攻击时硬件事件数据的获取。

算法1 HPCs事件数据采样算法

输出:进程标识符 *PID*,结构体 *attr*,采样次数 *T*。
输出:*attr* 对应性能事件次数 *count*。

```
1  i←1;
2  initialization(TYPE);//初始化 attr 结构体
3  read(PID);//读取 PID
4  fd←syscall(perf_event_open,attr);/*返回文件描述符*/
5  while i≤T do{
6    ioctl(fd,PERF_EVENT_IOC_RESET);/*复位性能计数器*/
7    ioctl(fd,PERF_EVENT_IOC_ENABLE);/*开启计数器*/
8    等待,采集对应进程的性能事件相关信息;
9    ioctl(fd,PERF_EVENT_IOC_DISABLE);/*停止计数*/
10   count←readcount(fd);/*读取对应的性能计数器值*/}
11 return count;
12 close(fd);//关闭计数器
```

3.2 特征分析

3.2.1 概述

计算机系统中HPCs的数量和记录的事件是有限的,但这并不代表监测全部HPCs的所有事件是一个好的检测方案。这是因为统计过多的事件信息会增加系统负载、影响检测精度;统计过少的事件信息无法准确刻画攻击特征,导致漏检增多、错检率上升。最小化监测事件集,使模型能在有限的统计资源中提升检测精度,降低系统性能开销,是本文需要解决的一个重要问题。

Wang等[25]提出应选取与攻击行为密切相关的事件来提高检测精度。Sabbagh等[38]认为应选取精确且独特的信息以增加攻击行为的辨识度。Chiappetta等[33]的研究发现,系统高负载时缓存替换频率和缓存未命中率上升,并且会产生对攻击检测不利的噪声,影响模型检测精度。因此,应该选择不同负载下对攻击特征影响较小的事件。机器学习实践表明,训练样本应尽量避免单一类型数据,可以减少模型偏向。

结合上述研究及本文的实验观察,本文认为可以从以下5个方面来选取事件集并从中选取分辨能力较强的特征来最小化监测事件集:

(1)事件对攻击行为的相关性。如缓存争用会使“缓存未命中”事件次数显著增加。

(2)事件的多样性。事件多样性有助于分类,缺少多样性的训练数据会造成模型的偏向。

(3)事件的鲁棒性。在系统负载增加的情况下,对特征影响小的事件。

(4)事件的独特性和精确性。尽可能提供精确、独特的信息,正常状态的信息和异常状态的信息不会与其他事件重叠。

(5)事件集的轻量性。少而精的事件集,采样时不会造成巨大的性能开销。

3.2.2 初步选择

从3.2.1节的5方面事件选取标准及攻击原理和具体实现出发,分析可能发生的硬件事件,有利于初步缩小监测事件集。本节从2种攻击实现代码及其原理出发,分析可能需要监测的硬件事件,并重点关注多层级的缓存事件信息。

Cache是为减小CPU运算速度与主存读写速度之间的差距而设计的高速存储器。当CPU访问的数据位于Cache中时,称为“缓存命中”(Cache Hit),相应地在HPCs中,该事件发生被“Cache References”记录。为便于后文叙述,本文将事件 *A* 与HPCs中性能事件 *B* 的对应关系记为〈*A*,*B*〉。即〈Cache Hit,Cache References〉。反之,则从主存读取数据并更新缓存,称为“缓存未命中”(〈Cache Miss,Cache Misses〉)。缓存命中与否会导致数据读取时间的显著差异,这种差异是Flush+

Reload 和 Prime+Probe 攻击的基础。

从事件对攻击行为的相关性与事件的多样性出发,在 Flush+Reload 攻击中,攻击进程会频繁调用 *clflush* 指令清空指定缓存行;在 Prime+Probe 攻击中,攻击者利用驱逐集将受害者的缓存行逐出缓存。这些行为都会导致本应在缓存中的数据需要从主存中读取。因此,理论上在这 2 种攻击发生时,被攻击系统的缓存未命中次数会高于正常系统的,监测 HPCs 上的 Cache References 和 Cache Misses 2 项指标对判断是否发生了攻击是非常必要的。

从事件的鲁棒性出发,除了关注 Cache 整体,本文还需要关注各层次缓存之间可能存在的差异。作为与处理器核心直接进行数据交互的 L1 Cache,数据缓存命中(〈L1 Data Cache Hit,L1 dCache Loads〉)与数据缓存未命中(〈L1 Data Cache Miss,L1 dCache Load Misses〉)事件能直接反映攻击行为。指令缓存的缓存命中情况(〈L1 Instruction Cache,Cache Instruction〉)以及未命中导致的程序执行时间(〈Program Execution Time,CPU Cycles〉)变长也是需要重点关注的信息,他们能在一定程度上刻画攻击行为。

L1 Cache 未命中时,会逐级检索下一级缓存,直至最后一级缓存 LLC(Last Level Cache)。Flush+Reload 和 Prime+Probe 是利用 LLC 的包容性进行信息窃取的侧信道攻击,因此,LLC 的命中(〈LLC Hit,L3 Total Accesses〉)与未命中(〈LLC Miss,L3 Total Cache Misses〉)也是本文重点监测的信息。

按所述步骤初步筛选出的硬件事件信息如表 1 所示,其中用事件名称首字母缩写表示事件发生的次数。

Table 1　Information of hardware events

表 1　相关硬件事件信息表

缩写事件名称	变量	事件信息
Cache Misses	*CM*	缓存丢失次数
Cache References	*CR*	缓存命中次数
CPU Cycles	*CC*	程序执行花费的 CPU 周期数
Cache Instruction	*CI*	指令读取次数
L1 dCache Loads	*L1d-CL*	L1 数据缓存读取次数
L1 dCache Load Misses	*L1d-CLM*	L1 数据缓存读取未命中次数
L3 Total Accesses	*L3-TA*	L3 全部访问次数
L3 Total Cache Misses	*L3-TCM*	L3 全部访问丢失次数

3.3　特征选择

从事件的独特性和精确性出发,为进一步缩小事件集,选出与攻击相关性较大的若干事件作为攻击特征。本文记录表 1 中的事件信息在攻击和无攻击场景下的相应值,并以折线图形式来可视化展现这些数值的变化情况,如图 2 和图 3 所示。图中横坐标表示采样次数,每次采样的时间间隔为 0.1 ms;纵坐标表示每次采样下各类事件发生的次数。

Prime+Probe 攻击发生时,表 1 所示的 8 个事件信息的变化情况如图 2 所示。从图 2a、图 2f、图 2g 和图 2h 可以看出,*CM*、*L1d-CLM*、*L3-TA*、*L3-TCM* 4 个特征的值均有了明显的上升。从图 2b 和图 2e 可以看出,*CR*、*L1d-CL* 有明显的下降。从图 2c 和图 2d 可以看出,*CC* 在前 55 次的采样有一定程度的上升,但在后续的采样中数据基本无变化;*CI* 的数据重叠度很高,在攻击前后基本无变化。

Flush+Reload 攻击发生时,表 1 所示的 8 个

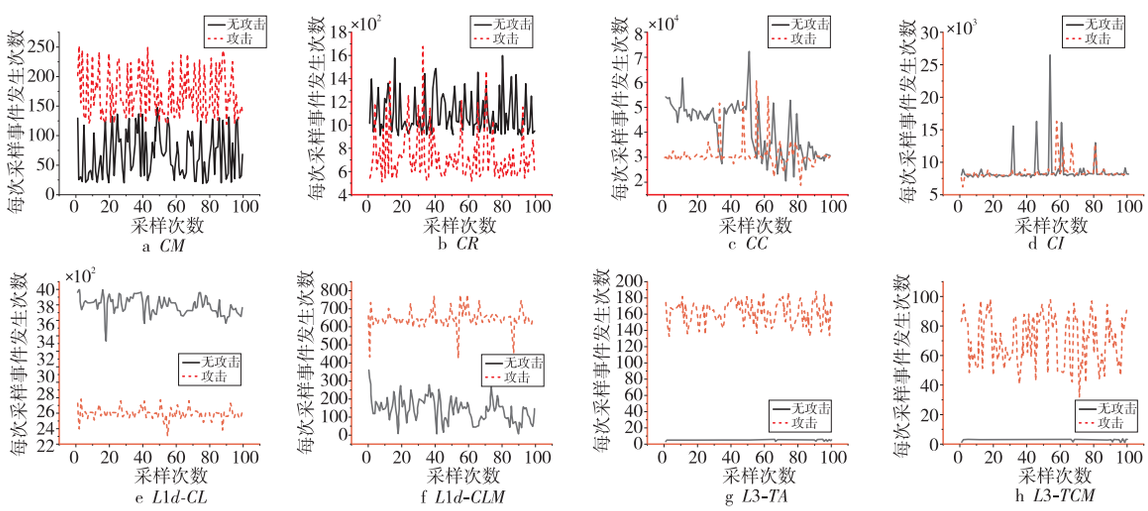

Figure 2　Changes of eight events information during Prime+Probe attack

图 2　Prime+Probe 攻击时 8 类事件信息的变化情况

事件信息的变化情况如图3所示。从图3a、图3f、图3g和图3h可以看出,*CM*、*L*1*d*-*CLM*出现了非常明显的增长;*L*3-*TA*和*L*3-*TCM*有了一定程度的增加,但增长幅度非常小,与Flush+Reload攻击相关性较低,几乎可以视为噪声。从图3b和图3e可以看出,*CR*、*L*1*d*-*CL*在Flush+Reload攻击下有了明显的降低。从图3c和图3d可以看出,*CI*、*CC*与在Prime+Probe攻击下的表现近似,图形重叠程度高,变化差异小,相关性低。

综上所述,在2种攻击下,各类事件信息都有不同程度的变化。其中相应值变化较大的有*CM*、*L*1*d*-*CLM*、*CR*,*L*1*d*-*CL*,此4类特征在2种攻击下,变化程度大、重叠程度低,具有较大的差异性。根据这4类特征可以有效判别攻击是否发生。*L*3-*TA*和*L*3-*TCM*这2类特征在Prime+Probe攻击下均出现了明显的增长,但在Flush+Reload攻击下这2类特征增长幅度小、相关性较低,所以不选择这2类事件信息作为主要特征。最后*CI*和*CC*特征在2类攻击下均表现出了较低的相关性。因此,针对Prime+Probe与Flush+Reload 2种攻击下的检测,本文选取*CM*、*CR*、*L*1*d*-*CLM*和*L*1*d*-*CL* 4个特征进一步构建数据集。

3.4 不同负载条件下特征信息的分析

本节分析和探究不同负载情况对4类特征信息的影响程度。对于检测模型部署的系统环境,系统高负载时相较于低负载时会出现更为频繁的缓存命中与未命中、换入与换出等事件,进而对以该事件及相关其他事件为特征的检测模型的检测精度造成一定影响。从事件集的轻量性出发,探究不同系统负载对特征信息的影响程度十分必要。3.3节选出了在Prime+Probe和Flush+Reload攻击下表现出高相关性的4类特征。基于这4类特征,本文引入*CMR*(Cache Misses Rates)和*L*1*d*-*CMR*(L1 data Cache Miss Rates)这2个可以反映出攻击行为与不同特征信息之间相关性的指标,并将其作为后续评估的衡量指标,具体计算分别如式(1)和式(2)所示:

$$CMR=\frac{CM}{CR} \tag{1}$$

$$L1d\text{-}CMR=\frac{L1d\text{-}CLM}{L1d\text{-}CL} \tag{2}$$

3.4.1 Flush+Reload

为了更好地对比不同负载情况下*CMR*和*L*1*d*-*CMR*变化情况,图4和图5分别给出了无系统负载和系统满载情况下Flush+Reload攻击发生时2个指标的变化情况。

从如图4可以看出,Flush+Reload攻击使得*CMR*和*L*1*d*-*CMR*在各采样批次中有了不同程度的增长,*CMR*平均提升了25%左右,*L*1*d*-*CMR*平均提升了35%左右,对总体采样批次而言,在Flush+Reload攻击下2个指标提升约37%。

从图5可以看出,无攻击发生时,*CMR*的区间明显较大,且在个别采样批次中*CMR*甚至超过了90%;Flush+Reload攻击发生时,*CMR*整体大于70%,有部分重叠区域;*L*1*d*-*CMR*在攻击和无攻击情况下所处的区间范围均较大,整体处于43%~80%,且有大部分重叠区域。

3.4.2 Prime+Probe

图6和图7分别是无系统负载和系统满载情况下,Prime+Probe攻击发生时*CMR*和*L*1*d*-

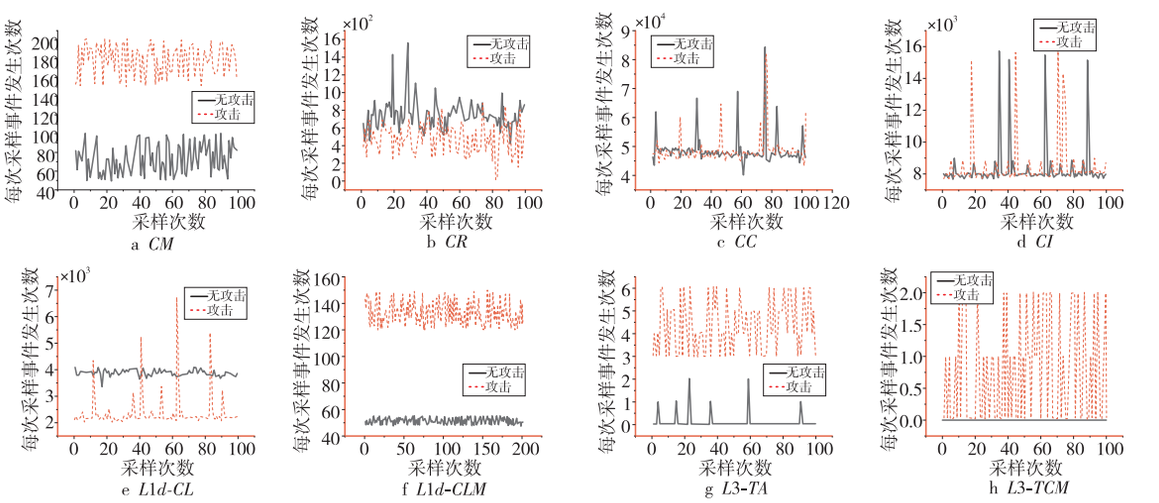

Figure 3 Changes of eight events information during Flush+Reload attack

图3 Flush+Reload攻击时8类事件信息的变化情况

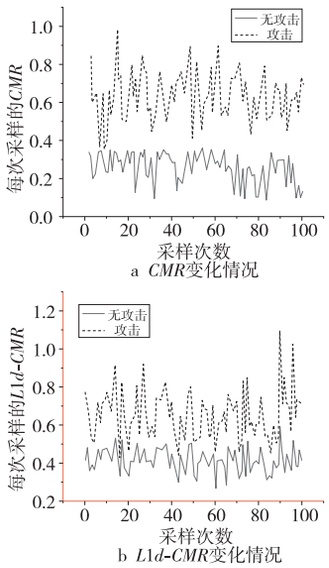

Figure 4　Changes of event information without system load (Flush+Reload)

图 4　无系统负载下事件信息的变化(Flush+Reload)

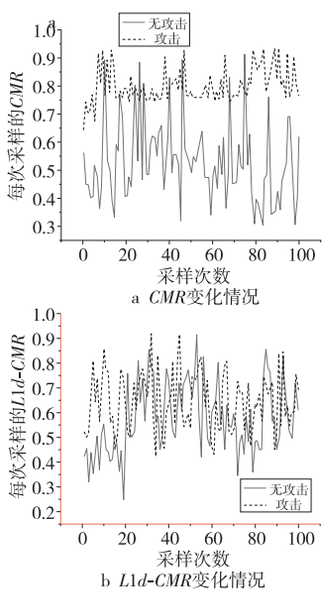

Figure 5　Changes of event information with fully system load (Flush+Reload)

图 5　系统满载下事件信息的变化(Flush+Reload)

CMR 变化情况。

从图 6 可以看出,Prime+Probe 攻击造成了大量的缓存丢失,*CMR* 整体在 70%以上,*L*1*d*-*CMR* 整体在 40%～80%;在无攻击情况下,*CMR* 整体小于 31%,*L*1*d*-*CMR* 在 10%～30%。

从图 7 可以看出,未受攻击时,*CMR* 基本处于 40%左右,相较于无负载情况提升了近 10%;在 Prime+Probe 攻击下,*CMR* 所处的区间范围有所增大,处于 40%以上,对比无系统负载情况时区间范围增大了约 30%;*L*1*d*-*CMR* 在攻击和无攻击

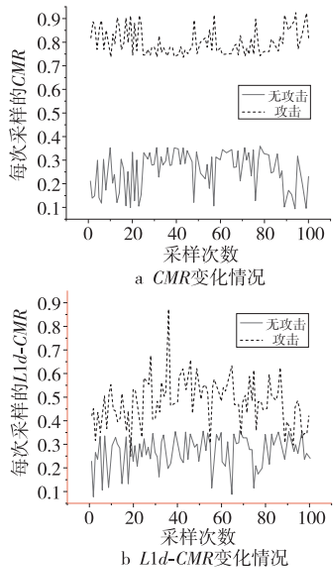

Figure 6　Changes of event information without system load (Prime+Probe)

图 6　无系统负载下事件信息的变化(Prime+Probe)

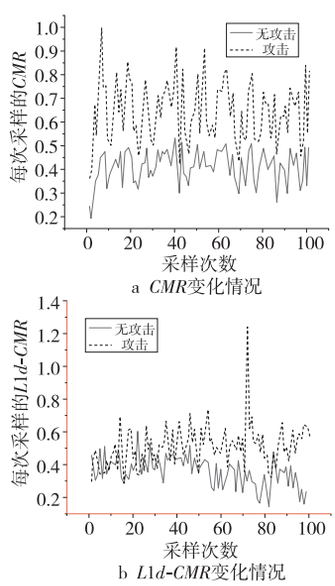

Figure 7　Changes of event information with fully system load (Prime+Probe)

图 7　系统满载下事件信息的变化(Prime+Probe)

下均集中在 40%左右,出现了部分重叠情况。

3.4.3　小结

从 3.4.1 节和 3.4.2 节的分析可知,攻击发生时,无系统负载情况下 *CMR* 和 *L*1*d*-*CMR* 2 个指标值均有不同程度的增长,图 4 和图 6 中 2 个指标基本无重叠区域,区分度明显;但系统满载时,2 个指标值都受到了不同程度的影响,数据所处的区间范围变大了,如图 5 和图 7 中 2 个指标出现了部分重叠。其中,*L*1*d*-*CMR* 受到的影响相对较大,重叠区域数据区分不明显,但还是有部分区域特征可

以用来判别攻击行为,直接忽视这部分数据会导致部分数据样本发生数值偏移或丢失。因此,在数据最终输入模型之前,还需要进行数据处理工作,将这一类远离了正常数据样本的离群点去除,通过特征放大和数据清洗等操作,使数据样本变得更加平滑,解决了数据的不一致性问题。图 8 展示了部分数据处理后的图形变化情况。

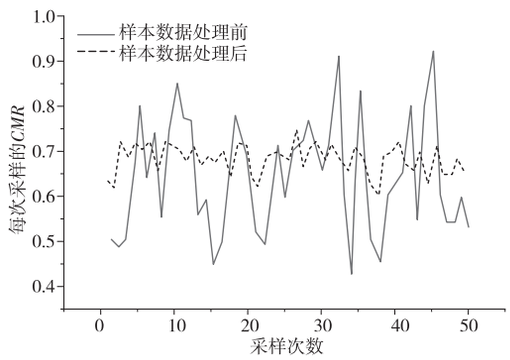

Figure 8 Example of samples changing before and after data processing

图 8 数据处理前后样本变化示例

数据处理完成后,数据集以用 4 个特征定义的 2 个指标为主要攻击特征,分别在有攻击和无攻击时,通过调用 *perf_event_open* 获取并统计无负载和系统满载时的特征数据,将不同系统环境下采集到的数据样本整合成最终所使用的数据集。最后将数据集划分成 3 个部分:训练集(占比 80%)、验证集(占比 10%)和测试集(占比 10%),分别用于模型训练、验证和测试。

4 DA 检测模型

传统检测模型在系统无负载时表现良好,特征区分较为明显,检测精度较高,响应速度较快。但是,在系统高负载时,各指标值均开始下降,难以满足实际检测需求。本节将阐述 DA 检测模型如何在高负载环境下保持高效的检测状态。

4.1 模型选择

根据 3.4 节的分析,系统无负载时,有攻击和无攻击的 *CMR* 和 *L*1*d*-*CMR* 有明显差异,可以用阈值方法进行区分。但是,在系统高负载时(如图 5b 和图 7b),缓存干扰导致部分数据重叠,阈值方法失效,只有少数特征数据能判别攻击。因此,需要构建能识别复杂模式的机器学习模型,用于探寻数据与攻击之间的规律,进行更有效的攻击识别。本节从检测精度和性能开销 2 方面评价不同的机器学习模型。

图 9 展示了不同系统负载下各类机器学习模型的检测精度和性能开销。检测精度如图 9a 所示,可以看到,随着系统负载增加,各机器模型的检测精度都有所下降,其中随机森林 RF(Random Forest)、*K* 近邻 KNN(*K*-Nearest Neighbors)、决策树 DT(Decision Tree)、神经网络 NNs(Neural Networks)和 AdaBoost 模型均表现出了不错的检测精度,并且在系统高负载的情况下 RF、NNs 和 AdaBoost 模型的检测精度均大于 90%。

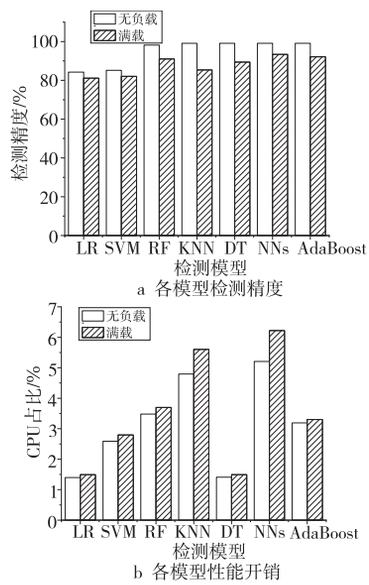

Figure 9 Detection accuracy and performance overhead of each machine learning model

图 9 各机器学习模型检测精度和性能开销

性能开销方面如图 9b 所示,可以看到,不同的检测模型性能开销也有所差异。其中逻辑回归 LR(Logistic Regression)和 DT 性能开销均小于 2%;支持向量机 SVM(Support Vector Machine)、RF 和 AdaBoost 模型的性能开销大约在 2%~5%;KNN 和 NNs 模型的性能开销在系统满载的情况下均超过了 5%。

通过综合对比分析可知,LR 和 SVM 的检测精度相对较低,不适合选用;KNN 模型检测精度较高,但它使用运行时的所有数据点来进行决策,性能开销和存储开销过大,实现复杂;NNs 模型检测精度也较高,但容易过拟合,性能开销较大;AdaBoost 模型在系统满载时的检测精度优于 RF 模型的;DT 不仅具有较低的性能开销,在系统无负载时也能保持不错的检测精度。综上,本文选取 DT 和 AdaBoost 构建新模型。

4.2 DA 检测模型设计与实现

4.2.1 DA 检测模型简介

(1)DA 检测模型的设计思想。基于 DT 和

AdaBoost 2 种模型各自在低负载和高负载情况下优异的检测性能,提出将 2 种检测模型组合使用的方法。该检测方法分为 D-A-C 3 个步骤。D 步骤选取决策树作为第 1 个分类器。为了更有效地划分数据集,在决策树分支节点中构造线性分类器来确定属性所属分类区间,通过递归生成决策树检测模型,用于完成低负载时的检测攻击。A 步骤选取 AdaBoost 作为第 2 个分类器。初始化数据样本权重,根据训练集中每个样本的分类正确性(即准确率),更新每个样本的权值和分布,通过反复迭代得到最终的强分类器,用于完成高负载时的检测攻击。C(Combination)步骤根据系统负载情况,动态调整上述 2 个模型的权重,然后输出最终的检测结果。

(2)训练策略。训练集中的数据样本均匀地分布在不同使用场景下,包含攻击和无攻击场景下的数据样本,以及无系统负载和系统满载情况下收集的数据样本。系统满载情况下,选取的特征数据会受到不同程度的影响,而这一类数据往往是造成检测精度降低和误报率升高的关键因素。因此,为了使得检测模型具有更好的检测效果,本文在数据集中将无系统负载和系统满载的数据样本分开,先使用无系统负载下的数据样本进行预训练,并保留预训练的参数;然后加入系统满载数据样本,根据参数对比调整,使得训练模型达到一个好的检测效果。

4.2.2　决策树检测模块的构建

数据集中每个样本对应于坐标空间中的一个数据点。这种情况下,样本的分类可以被视为在坐标空间中寻找分类边界。然而,决策树的一个显著特点是决策节点常使用属性的某一阈值作为分类依据,这一特点在坐标空间中表现为分类边界与坐标轴平行。尽管这种划分方式具有很好的解释性,但在处理复杂分类样本时,通常需要逐个经过多个决策节点才能获得较好的决策效果,这会导致决策树变得相当复杂。本文通过一条斜线来划分边界,解决了决策树这方面的问题。此时,决策树的每个非叶子节点都可以看作是一个线性分类器,通过调整线性分类器的参数,可以获得更好的检测效果。具体而言,主要步骤如下:

(1)导入训练集和属性集:选取 *CMR* 和 *L*1*d*-*CMR* 为构建数据集的主要特征,由此确定了训练决策树的数据样本和属性样本。其中,数据样本为攻击和无攻击时采样的数据集合,属性样本为数据集特征集合。

(2)构建分支节点:对决策树的每个分支节点构造线性分类器,以 *CMR* 和 *L*1*d*-*CMR* 分别作为 *X* 轴和 *Y* 轴构建坐标系。如图 10 所示,坐标系中标记了训练集中的部分数据点,三角形点为攻击下的数据点,方形点为无攻击下的数据点。在训练样本的每个属性数据样本均表示为坐标平面上的点后,通过最小二乘法在平面中找到一条直线,使得所有样本到该直线的距离之和最小,且该斜线能区分正常数据和攻击数据。

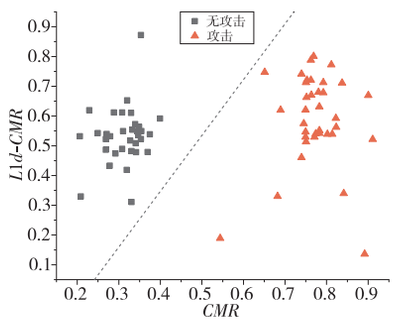

Figure 10　Linear classification model

图 10　线性分类模型

(3)递归构建决策树:将选取好的特征用相同的方式进行属性划分,同时定义决策树节点。此时,决策树中每个非叶节点都是一个线性分类器,用于表示已经对属性进行了划分,叶节点则代表了最终的决策结果。每一次属性划分后,数据被传递到下一个节点,并通过递归的方式来构建整棵决策树。当遍历完所有的属性样本时递归过程结束,产生了完整的决策树。如图 11 所示为最终递归完成的多变量决策树。

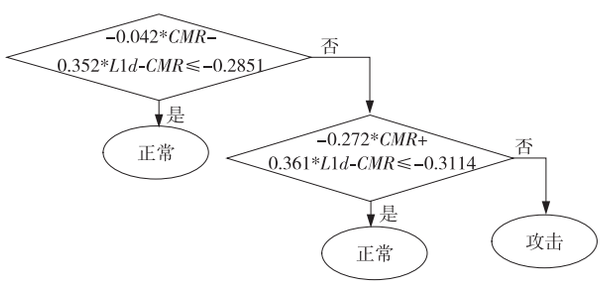

Figure 11　Example of multivariate decision tree completed through recursion

图 11　递归完成的多变量决策树示例

4.2.3　AdaBoost 检测模块的构建

为保证高负载时检测模型的可靠性和精度,引入 AdaBoost 检测模块。如图 12 所示为 AdaBoost 训练流程,通过更新样本权重和分布以及调整迭代次数,构建强分类器。

主要步骤如下:

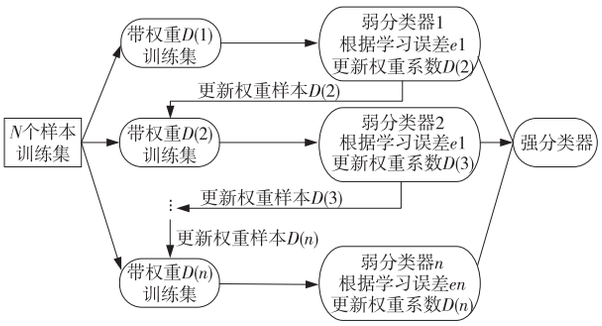

Figure 12 Training process of AdaBoost

图 12 AdaBoost 训练流程

(1)导入训练样本并选择弱分类器:如表 3 所示,在数据集中随机抽取 1 000 条数据作为初始训练样本,并为每个数据样本分配 0.001 的初始权重。使用线性分类器作为弱分类器。

(2)更新权重和样本分布:利用线性分类器对训练集进行划分,并调整线性分类器的阈值,以使不同标签的数据点能够被线性分类器分开。然后,统计分类错误的数据样本。表 2 后 3 行显示了 1 次训练的分类结果以及相应的样本权重变化。正确样本的权重从 0.001 减少到 0.000 67,错误样本的权重从 0.001 提升到 0.001 8。本文计算了线性模型的初始误差率,即分类错误的数据点个数占数据集中所有数据点的比例。根据误差率,更新每个样本的权重和样本分布,以进行下一轮迭代。

Table 2 Samples, classification results and weight updating

表 2 样本、分类结果及更新权重

	样本 1	样本 2	样本 3	…	样本 1 000
CMR	0.24	0.82	0.19	…	0.90
*L*1*d*-*CMR*	0.28	0.71	0.35	…	0.68
类别标签	0	1	0	…	1
分类器结果	0	1	0	…	0
分类结果	对	对	对	…	错
初始权重	0.001	0.001	0.001	…	0.001
更新权重	0.000 67	0.000 67	0.000 67	…	0.001 8

(3)迭代构成强分类器:第 1 轮对样本数据加权更新后,AdaBoost 检测模型已初步具备了检测能力。为进一步降低误差率,本文重复上述过程进行多轮迭代,不断生成新的弱分类器,并更新样本权重和分布。AdaBoost 检测模型的迭代次数由预先设定的训练轮数 n 决定。通过调整训练轮数,可以使得 AdaBoost 检测模型获得较高的检测精度。需要指出的是,这并不代表迭代次数与模型性能成正相关。若迭代次数太少,模型训练不充分,导致检测精度低;迭代次数过多,易导致模型训练耗时长,甚至出现过拟合问题。如图 13 所示,本文 AdaBoost 迭代到 100 次时检测精度最高。因此,本文将迭代次数设置为 100,用于后续训练和测试。图 14 是根据 AdaBoost 检测模型迭代原理做出的一种图形化解释,每次权重更新都可以从样本中划分出更多的有效数据点,最终通过迭代获得数据的有效划分。

AdaBoost	
迭代次数	检测精度/%
1	74.48
10	81.35
50	94.73
100	99.95
200	99.72
500	98.12

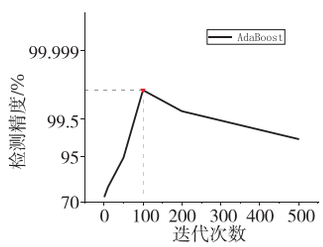

Figure 13 Relationship between the number of iterations and detection accuracy of AdaBoost

图 13 AdaBoost 迭代次数与检测精度关系

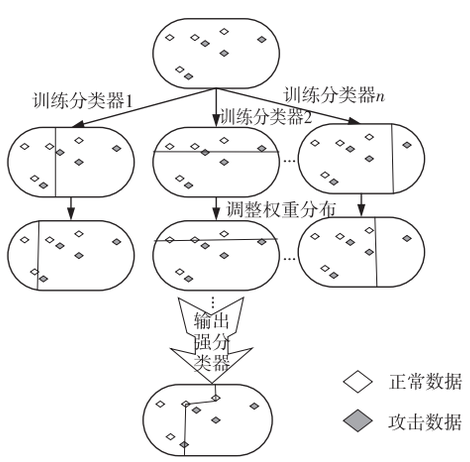

Figure 14 Training process of AdaBoost

图 14 AdaBoost 训练流程

4.2.4 模型的组合

本文分别给决策树和 AdaBoost 模型设置一个初始权重 $w_{\mathrm{DT}}^{(0)}$ 和 $w_{\mathrm{Ada}}^{(0)}$。2 个权重的关系始终满足式(3)。其中,i 表示迭代次数,0 表示初始值。如式(4)所示,通过一个模型选择函数 $\phi(\cdot)$ 随机选取每次用于检测的模型,模型被选中的概率与模型权重呈正相关。权重根据系统负载动态变化,且与检测精度、响应时间相关。系统负载较小时,可增大 $w_{\mathrm{DT}}^{(i)}$,以保持较快的检测速度和较高的检测精度;系统负载较大时,可增大 $w_{\mathrm{Ada}}^{(i)}$,以保证模型在

高负载时具有较高的检测精度。

$$w_{\mathrm{DT}}^{(i)}+w_{\mathrm{Ada}}^{(i)}=1 \tag{3}$$

$$\phi\left(w_{\mathrm{DT}}^{(i)},w_{\mathrm{Ada}}^{(i)}\right)=\begin{cases}Model_{\mathrm{DT}},\text{低系统负载时}\\ Model_{\mathrm{AdaBoost}},\text{高系统负载时}\end{cases} \tag{4}$$

5　实验评估

本文在 Intel i7-4770 CPU 上运行 Linux Ubuntu16.04.1 进行实验。本节将采用多种机器学习模型和不同负载条件下 DA 检测模型的性能进行对比。5.1 节比较各种机器学习模型在针对缓存侧信道攻击检测的二分类问题中的效果,并从中选取出最优模型作为 DA 检测模型的基础。5.2 节分析 DA 检测模型在无负载和满载情况下与其他机器学习模型在各项性能指标上的差异。

5.1　机器学习模型实验对比

本文选取了决策树和 AdaBoost 2 种机器学习模型构建检测模型,并测试了其相关性能指标,同时还选择了 SVM 和 RF 作为对比模型。本节从检测精度、检测时间、误报率和性能开销 4 个方面评价模型的检测效果。检测精度是指检测模型预测结果与实际结果的一致性。检测速度是指当攻击发生到被检测出的时间间隔。误报率在检测中主要分为假阳性率和假阴性率,其中假阳性是指系统在正常状态下被预测为攻击;假阴性是指系统在攻击状态下被检测为正常。误报率可以通过统计错误结果在总结果中的占比来计算。性能开销是指运行检测模型时的 CPU 占用率。

表 3 展示了 4 种针对侧信道攻击检测的机器学习模型在二分类问题中的实验结果,分别考虑了 Prime+Probe 和 Flush+Reload 2 种攻击在无系统负载和系统满载的情况下的检测效果。

从图 5 可以看出,无系统负载时,决策树和 AdaBoost 相较于 SVM 与 RF 具有更高的检测精度和更低的误报率,但是相比于其他 3 种模型,AdaBoost 的性能开销明显较高,此时选用决策树作为检测模型是兼顾精度和开销的最优选择;系统满载时,AdaBoost 在检测精度和误报率方面的优势更加突出;RF 在无负载和满载情况下响应时间都较长,不符合模型的响应时间要求; SVM 在无负载和满载情况下的误报率都较高,不符合模型的准确率要求;DT 检测精度较低。因此,选用 AdaBoost 作为检测模型是兼顾精度和开销的最优选择。

5.2　DA 检测模型性能评估

本节将对 DA 检测模型的各项指标进行评估。表 4 为无系统负载和系统满载情况下 DA 检测模型各项指标对比情况。

(1)检测精度:从表 4 可以看出,DA 检测模型在 Prime+Probe 攻击下的检测精度在 98.85%～99.95%;在 Flush+Reload 攻击下的检测精度在 98.78%～99.98%。无系统负载情况下,DA 检测模型的检测精度最高达到了 99.98%,系统满载情况下,检测精度有所降低,在 98.80%左右。

对比 Prime+Probe 攻击下的单个检测模型:无负载情况下,DA 检测模型的检测精度比 DT、SVM、RF 的均提升了 0.4%左右;满载情况下,单个模型与 DA 检测模型的检测精度均有了一定的提升,与 AdaBoost 模型的相当。

对比 Flush+Reload 攻击下的单个检测模型:无负载情况下,DA检测模型检测精度相比DT、

Table 3　Experimental results of single model under Prime+Probe (PP) and Flush+Reload (FR) attacks

表 3　Prime+Probe 与 Flush+Reload 攻击下单个模型实验结果

模型	系统负载	检测精度/%		响应时间/ms		误报率/%		性能开销/%	
		Prime+Probe	Flush+Reload	Prime+Probe	Flush+Reload	Prime+Probe	Flush+Reload	Prime+Probe	Flush+Reload
DT	无负载	99.88	99.86	15	15	0.12	0.12	1.4	1.4
	满载	96.54	96.78	17	17	3.46	3.22	1.5	1.5
SVM	无负载	99.07	97.54	14	14	0.93	2.46	2.6	2.6
	满载	97.01	95.82	20	15	2.99	4.18	2.8	2.8
RF	无负载	99.72	99.92	21	20	0.28	0.08	2.1	2.1
	满载	97.73	97.36	30	28	2.27	2.64	2.3	2.3
AdaBoost	无负载	99.95	100	17	16	0.05	0	3.2	3.2
	满载	98.84	98.76	23	23	0.26	0.24	3.2	3.2

Table 4 Experimental results of DA detection model under different load conditions

表 4 不同负载情况下 DA 检测模型实验结果

系统负载	检测精度/%		响应时间/ms		误报率/%		性能开销/%	
	Prime+Probe	Flush+Reload	Prime+Probe	Flush+Reload	Prime+Probe	Flush+Reload	Prime+Probe	Flush+Reload
无负载	99.95	99.98	15	15	0.05	0.02	3.5	3.5
满载	98.85	98.78	20	22	1.35	1.22	3.7	3.7

SVM、RF 的有小幅度提升,但略小于 AdaBoost 模型的;满载情况下,DA 检测模型的检测精度略小于 AdaBoost 模型的,相比 DT、SVM、RF 的检测精度提升约 3%。

(2)检测速度:DA 检测模型在 Prime+Probe 攻击和 Flush+Reload 攻击下的响应时间为 15~22 ms。其中,无负载情况下,DA 检测模型的检测时间最小,只有 15 ms;满负载情况下,DA 检测模型的最高,有 22 ms。

对比 DT 和 SVM,系统无负载情况下,DA 检测模型的检测速度相当。但在系统满载情况下,DA 检测模型的响应时间明显长于 DT 和 SVM 的,略短于 AdaBoost 模型的,相比 RF 的检测速度要快约 7 ms。

(3)误报率:DA 检测模型在 Prime+Probe 和 Flush+Reload 攻击中表现出了较好的检测效果,其误报率相对较低,均小于 0.1%。但是,在系统满载的情况下,仍有 1.3%左右的误报率。对比 DT、SVM 和 RF 这 3 个检测模型,DA 检测模型的误报率均小于这 3 个模型的。DA 检测模型的误报率与 AdaBoost 的相当。

(4)系统性能开销:无负载情况下,DA 检测模型在 Prime+Probe 和 Flush+Reload 攻击时的 CPU 占比为 3.5%,且随着系统负载的增加,DA 检测模型的性能开销也略微增长了 0.2%。与 4 个单个检测模型相比,DA 检测模型检测的性能开销与 AdaBoost 的相当,也可以看出 DA 检测模型的性能开销更趋向于 AdaBoost 模型的。

实验结果表明,在系统无负载与满载情况下,DA 检测模型相较于 4 种模型具有更高的检测精度、更快的检测速度和更低的误报率。性能开销略高于其他 3 种模型的,相较于 AdaBoost 的高约 0.2%,相较于其他方面的改进,性能开销在可接受范围内。

6 结束语

针对计算机系统面临缓存侧信道攻击的要求检测具备实时性、检测精度高,且检测情况更复杂等问题,本文提出了一种基于决策树和 AdaBoost 的 DA 检测模型。同时,本文研究了硬件事件特征在不同负载和不同攻击情况下的变化规律。DA 检测模型综合考虑了检测速度、检测精度和系统性能开销等因素,有效地提高了计算机系统对缓存侧信道攻击的检测能力。实验结果表明,在不同负载和攻击情况下,DA 检测模型的性能优于传统的单个检测模型的,具有更高的检测精度和更低的误报率,能够在不高于 3.7%的系统开销下达到 98.8%的平均检测精度,满足了不同负载情况下攻击检测的实时性、准确性要求,为缓存侧信道攻击的实时检测提供了有效的解决方案。

参考文献:

[1] Kocher P C. Timing attacks on implementations of Diffie-Hellman,RSA,DSS,and other systems[C]//Proc of the 16th Annual International Cryptology Conference,1996:104-113.

[2] Kocher P,Jaffe J,Jun B. Differential power analysis[C]//Proc of the 19th Annual International Cryptology Conference,1999:388-397.

[3] Mangard S. A simple power-analysis (SPA) attack on implementations of the AES key expansion[C]//Proc of the 5th International Conference on Information Security and Cryptology,2003:343-358.

[4] Brier E,Clavier C,Olivier F. Correlation power analysis with a leakage model[C]//Proc of the 6th International Conference on Cryptographic Hardware and Embedded Systems,2004:16-29.

[5] Bernstein D J. Cache-timing attacks on AES [EB/OL]. [2023-05-23]. https://mimoza.mamara.edu.tr/~mskalli/cse466_09/cache timing-20050414.pdf.

[6] Hund R,Willems C,Holz T. Practical timing side channel attacks against kernel space ASLR[C]//Proc of 2013 IEEE Symposium on Security and Privacy,2013:191-205.

[7] Osvik D A,Shamir A,Tromer E. Cache attacks and countermeasures:The case of AES[C]//Proc of the Cryptographers' Track at the RSA Conference,2006:1-20.

[8] Zhang Y,Juels A,Reiter M K,et al. Cross-VM side channels and their use to extract private keys[C]//Proc of the 2012 ACM Conference on Computer and Communications Security,2012:305-316.

[9] Yarom Y, Genkin D, Heninger N. CacheBleed:A timing at-

tack on OpenSSL constant-time RSA[J]. Journal of Cryptographic Engineering,2017,7:99-112.

[10] Kocher P,Horn J,Fogh A,et al. Spectre attacks:Exploiting speculative execution[J]. Communications of the ACM,2020,63(7):93-101.

[11] Lipp M,Schwarz M,Gruss D,et al. Meltdown:Reading kernel memory from user space[J]. Communications of the ACM,2020,63(6):46-56.

[12] van Bulck J, Minkin M, Weisse O, et al. Foreshadow: Extracting the keys to the Intel SGX kingdom with transient out-of-order execution[C]//Proc of the 27th USENIX Conference on Security Symposium,2018:991-1008.

[13] Chen G X,Chen S C,Xiao Y,et al. SgxPectre:Stealing Intel secrets from SGX enclaves via speculative execution[C]// Proc of 2019 IEEE European Symposium on Security and Privacy,2019:142-157.

[14] Su C,Zeng Q. Survey of CPU cache-based side-channel attacks: Systematic analysis, security models, and countermeasures[J]. Security and Communication Networks,2021:1-15.

[15] Gruss D,Maurice C,Wagner K,et al. Flush+Flush:A fast and stealthy cache attack[C]//Proc of the 13th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment,2016:279-299.

[16] Yarom Y, Falkner K. Flush+Reload: A high resolution, low noise, L3 cache side-channel attack[C]//Proc of the 23rd USENIX Security Symposium, 2014:719-732.

[17] Liu F,Yarom Y,Ge Q,et al. Last-level cache side-channel attacks are practical[C]//Proc of 2015 IEEE Symposium on Security and Privacy,2015:605-622.

[18] Briongos S,Malagón P,Moya J M,et al. Reload+Refresh: Abusing cache replacement policies to perform stealthy cache attacks[C]//Proc of the 29th USENIX Conference on Security Symposium,2020:1967-1984.

[19] Mushtaq M,Akram A,Bhatti M K,et al. Nights-watch:A cache-based side-channel intrusion detector using hardware performance counters[C]//Proc of the 7th International Workshop on Hardware and Architectural Support for Security and Privacy,2018:1-8.

[20] Allaf Z,Adda M,Gegov A. A comparison study on Flush+Reload and Prime+Probe attacks on AES using machine learning approaches[C]//Proc of the 17th UK Workshop on Computational Intelligence:Advances in Computational Intelligence Systems,2018:203-213.

[21] Briongos S,Irazoqui G,Malagón P,et al. CacheShield:Detecting cache attacks through self-observation[C]//Proc of the 8th ACM Conference on Data and Application Security and Privacy,2018:224-235.

[22] Tong Z,Zhu Z,Wang Z,et al. Cache side-channel attacks detection based on machine learning[C]//Proc of 2020 IEEE 19th International Conference on Trust,Security and Privacy in Computing and Communications,2020:919-926.

[23] Kulah Y, Dincer B, Yilmaz C, et al. SpyDetector: An approach for detecting side-channel attacks at runtime[J]. International Journal of Information Security, 2019, 18: 393-422.

[24] Demme J,Maycock M,Schmitz J,et al. On the feasibility of online malware detection with performance counters[C]// Proc of the 40th Annual International Symposium on Computer Architecture,2013:559-570.

[25] Wang H, Sayadi H, Kolhe G, et al. Phased-guard: Multiphase machine learning framework for detection and identification of zero-day microarchitectural side-channel attacks [C]//Proc of 2020 IEEE 38th International Conference on Computer Design,2020:648-655.

[26] Ou Y, Li L. Side-channel analysis attacks based on deep learning network[J]. Frontiers of Computer Science,2022,16:10-11.

[27] Gruss D,Spreitzer R,Mangard S. Cache template attacks: Automating attacks on inclusive last-level caches[C] //Proc of the 24th USENIX Security Symposium,2015:897-912.

[28] Le A T,Hoang T T,Dao B A,et al. A real-time cache side-channel attack detection system on RISC-V out-of-order processor[J]. IEEE Access,2021,9:164597-164612.

[29] Sangeetha G,Sumathi G. An optimistic technique to detect cache based side channel attacks in cloud[J]. Peer-to-Peer Networking and Applications,2021,14:2473-2486.

[30] Zhang T,Zhang Y,Lee R B. Analyzing cache side channels using deep neural networks[C]//Proc of the 34th Annual Computer Security Applications Conference,2018:174-186.

[31] Gnanavel S,Narayana K E,Jayashree K,et al. Implementation of block-level double encryption based on machine learning techniques for attack detection and prevention[J]. Wireless Communications and Mobile Computing,2022(2):21-30.

[32] Oshana R,Thornton M A,Caraman M. A side channel attack detection system using processor core events and a support vector machine[C]//Proc of 2022 11th Mediterranean Conference on Embedded Computing,2022:1-8.

[33] Chiappetta M, Savas E, Yilmaz C. Real time detection of cache-based side-channel attacks using hardware performance counters[J]. Applied Soft Computing,2016,49:1162-1174.

[34] Irazoqui G,Inci M S,Eisenbarth T,et al. Know thy neighbor:Cryptolibrary detection in cloud[J]. Proceedings on Privacy Enhancing Technologies,2015,2015:25-40.

[35] Zhang T W, Zhang Y Q,Lee R B. CloudRadar:A real-time side-channel attack detection system in clouds[C]//Proc of the 19th International Symposium on Research in Attacks, Intrusions,and Defenses,2016:118-140.

[36] Shusterman A,Kang L,Haskal Y,et al. Robust website fingerprinting through the cache occupancy channel[C]//Proc of the 28th USENIX Security Symposium,2019:639-656.

[37] Chen J, Venkataramani G. CC-Hunter: Uncovering covert timing channels on shared processor hardware[C]//Proc of

the 47th Annual IEEE/ACM International Symposium on Microarchitecture,2014:216-228.

[38] Sabbagh M,Fei Y S,Wahl T,et al. SCADET:A side-channel attack detection tool for tracking prime-probe[C]//Proc of 2018 IEEE/ACM International Conference on Computer-Aided Design,2018:1-8.

[39] Bazm M-M,Sautereau T,Lacoste M,et al. Cache-based side-channel attacks detection through Intel cache monitoring technology and hardware performance counters[C]//Proc of 2018 3rd International Conference on Fog and Mobile Edge Computing,2018:7-12.

[40] Wang Z, Taram M, Moghimi D, et al. NVLeak: Off-chip side-channel attacks via non-volatile memory systems[C]// Proc of the 32nd USENIX Security Symposium,2023:6771-6788.

[41] Cook J,Drean J,Behrens J,et al. There's always a bigger fish: A clarifying analysis of a machine-learning-assisted side-channel attack[C]//Proc of the 49th Annual International Symposium on Computer Architecture, 2022: 204-217.

作者简介:

李扬(1999 -),男,江西上饶人,硕士生,CCF 会员(P2010G),研究方向为缓存侧信道攻击与防御。**E-mail**:1136019639@qq.com

LI Yang,born in 1999,MS candidate,CCF member(P2010G),his research interest includes cache side channel attack & defense.

尹大鹏(1994 -),男,宁夏银川人,硕士,研究方向为计算机系统安全。**E-mail**:993027110@qq.com

YIN Da-peng,born in 1994,MS,his research interest includes computer system security.

马自强(1990 -),男,新疆乌鲁木齐人,博士,副教授,CCF 会员(E6270M),研究方向为计算机系统安全和区块链应用安全。**E-mail**:maziqiang@nxu.edu.cn

MA Zi-qiang,born in 1990,PhD,associate professor,CCF member(E6270M),his research interests include computer system security and blockchain application security.

姚梓豪(1999 -),男,安徽阜阳人,硕士生,CCF 会员(P2843G),研究方向为计算机系统安全。**E-mail**:759760084@qq.com

YAO Zi-hao,born in 1999,MS candidate,CCF member(P2843G),his research interest includes computer system security.

魏良根(1997 -),男,四川成都人,硕士生,CCF 会员(P2845G),研究方向为格攻击。**E-mail**:122521514@qq.com

WEI Liang-gen,born in 1997,MS candidate, CCF member (P2845G), his research interest includes lattice attack.