Beijing Forest Studio
北京理工大学信息系统及安全对抗实验中心

虚拟化云平台异常行为检测方法

博士研究生　张毅飞

2022年05月08日

内容提要

- 背景简介
- 基本概念
  - 云计算平台
  - 虚拟化技术
  - 云安全威胁模型
  - 虚拟机数据源
- 算法原理
  - **VMGuard**
  - **LogkeyDCG**
  - 序列数据转换
- 应用总结
- 参考文献

预期收获

- 预期收获
  - 1. 了解虚拟化云平台面临的安全威胁与挑战
  - 2. 了解可用于云平台虚拟机安全检测的系统数据源
  - 3. 理解利用系统调用序列、系统运行日志的虚拟机异常检测方法
  - 4. 了解多种序列数据转换方法
  - 5. 了解应用领域和发展方向等

背景简介

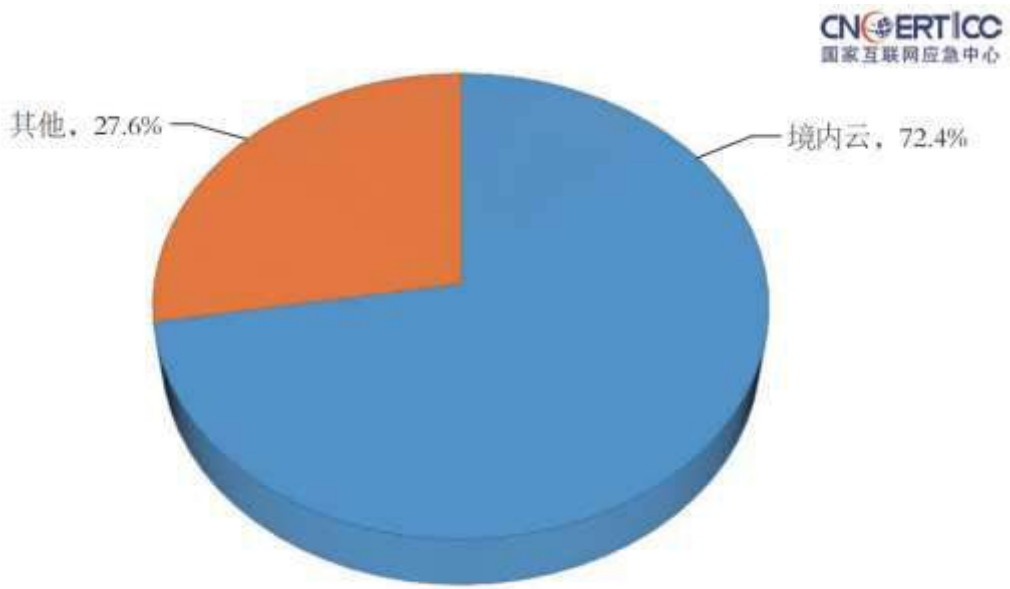
- 云计算平台通过高效便捷地向用户提供云服务来提高计算基础设施利用率
  - 云服务已经被许多组织广泛采用
- 快捷、丰富的服务交付模式
  - **SaaS (software as a service)** 软件即服务，如虚拟桌面，在线游戏等
  - **PaaS (platform as a service)** 平台即服务，如开发工具，**web**服务等
  - **IaaS (infrastructure as a service)** 基础架构即服务，虚拟机，服务器等
- 虚拟化在云计算的构建中起着至关重要的作用
  - 虚拟化技术允许多个操作系统和应用程序在物理基础设施之上运行
  - 形成分布式、共享、动态和多租户的体系结构

背景简介

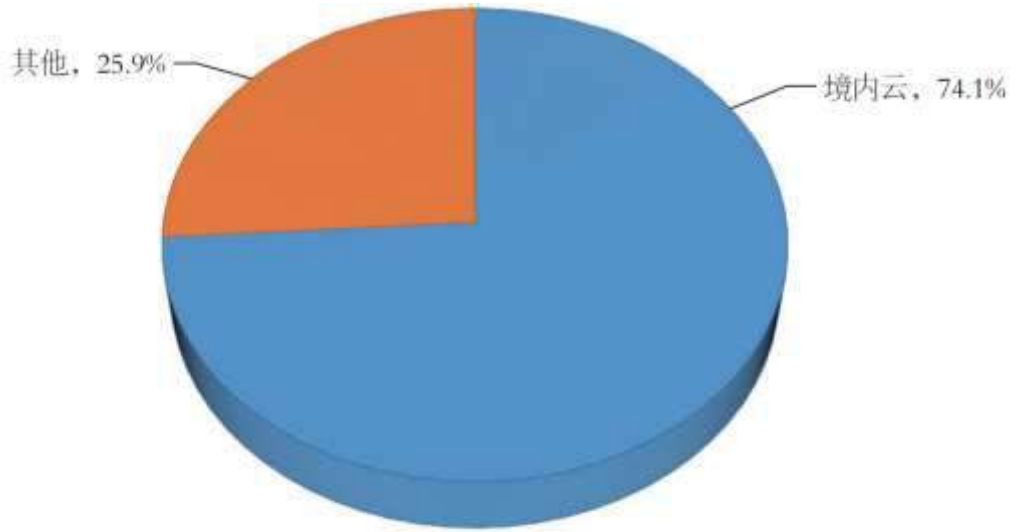
- 虚拟化云平台对信息化社会发展具有强大推动作用
  - 云类型：公有云、私有云、混合云等
  - 行业云：政务云、教育云、医疗云等
  - 商业云：阿里云、腾讯云、百度云、华为云等
- 总体市场规模稳步扩大
- 中国信息通信研究院在2021年发布的《云计算白皮书》中指出
  - 全球云计算市场规模为2083亿美元
  - 我国云计算整体市场规模也达到了2091亿人民币

背景简介

- 虚拟化云平台仍然面临巨大的安全威胁和挑战
  - 《2020年中国互联网网络安全报告》指出，与2019年相比，我国共有38779个境内云主机受木马或僵尸网络程序控制，大量被控主机被用于发起其他攻击



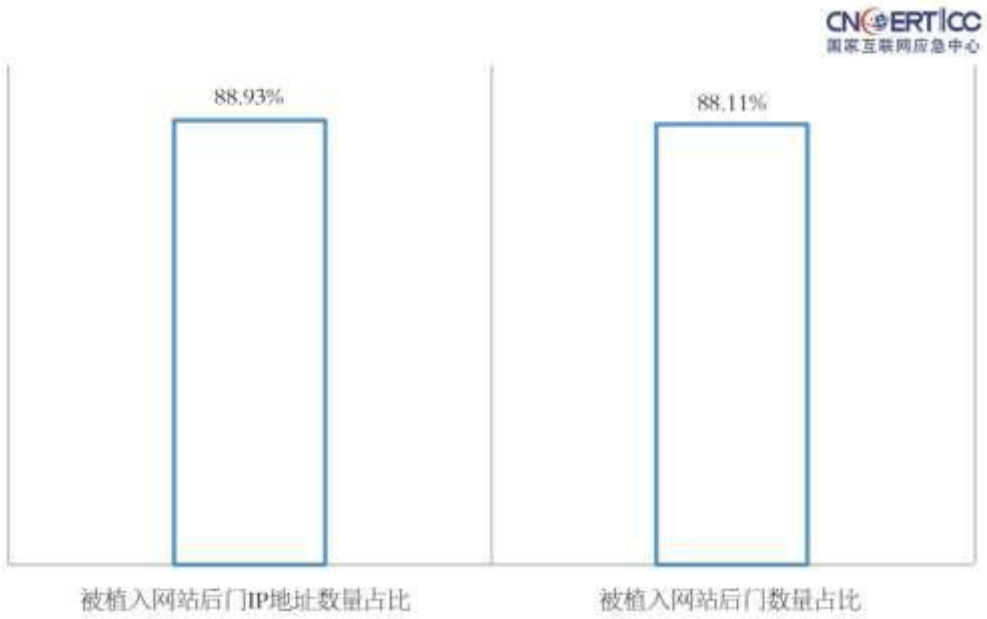
（a）遭受大流量DDoS攻击IP地址数量占比

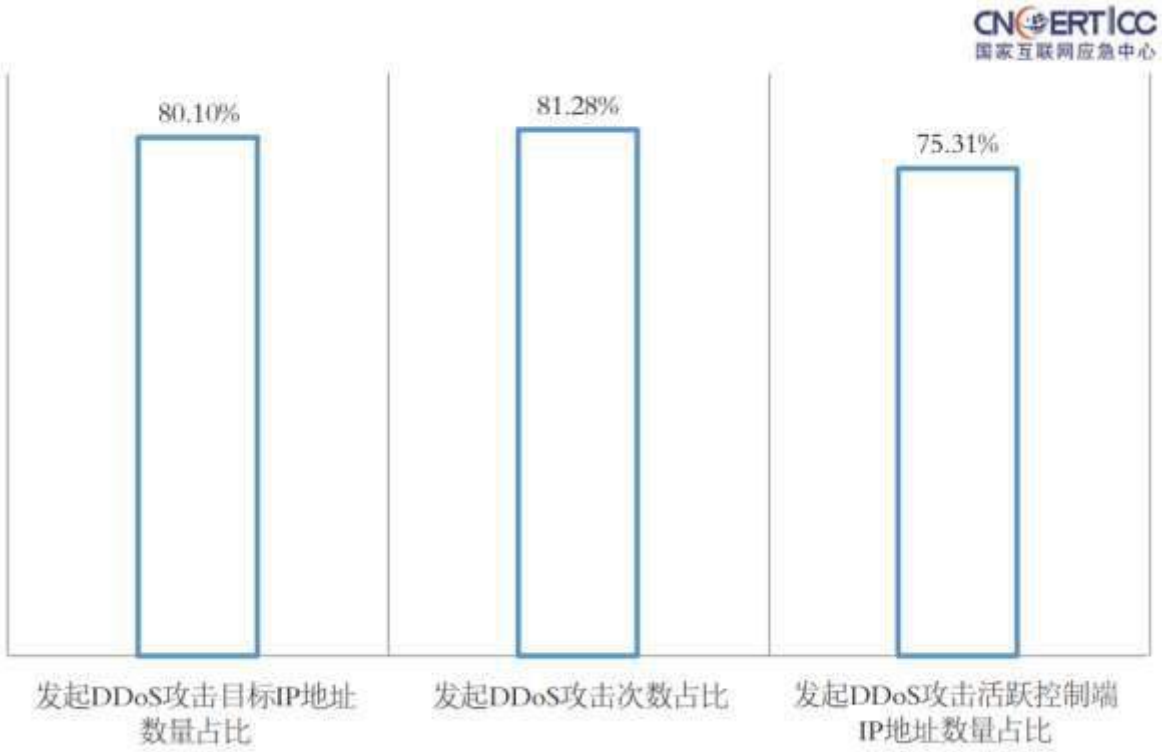


（b）遭受大流量DDoS攻击次数占比

背景简介

- 网络攻击的重灾区





背景简介

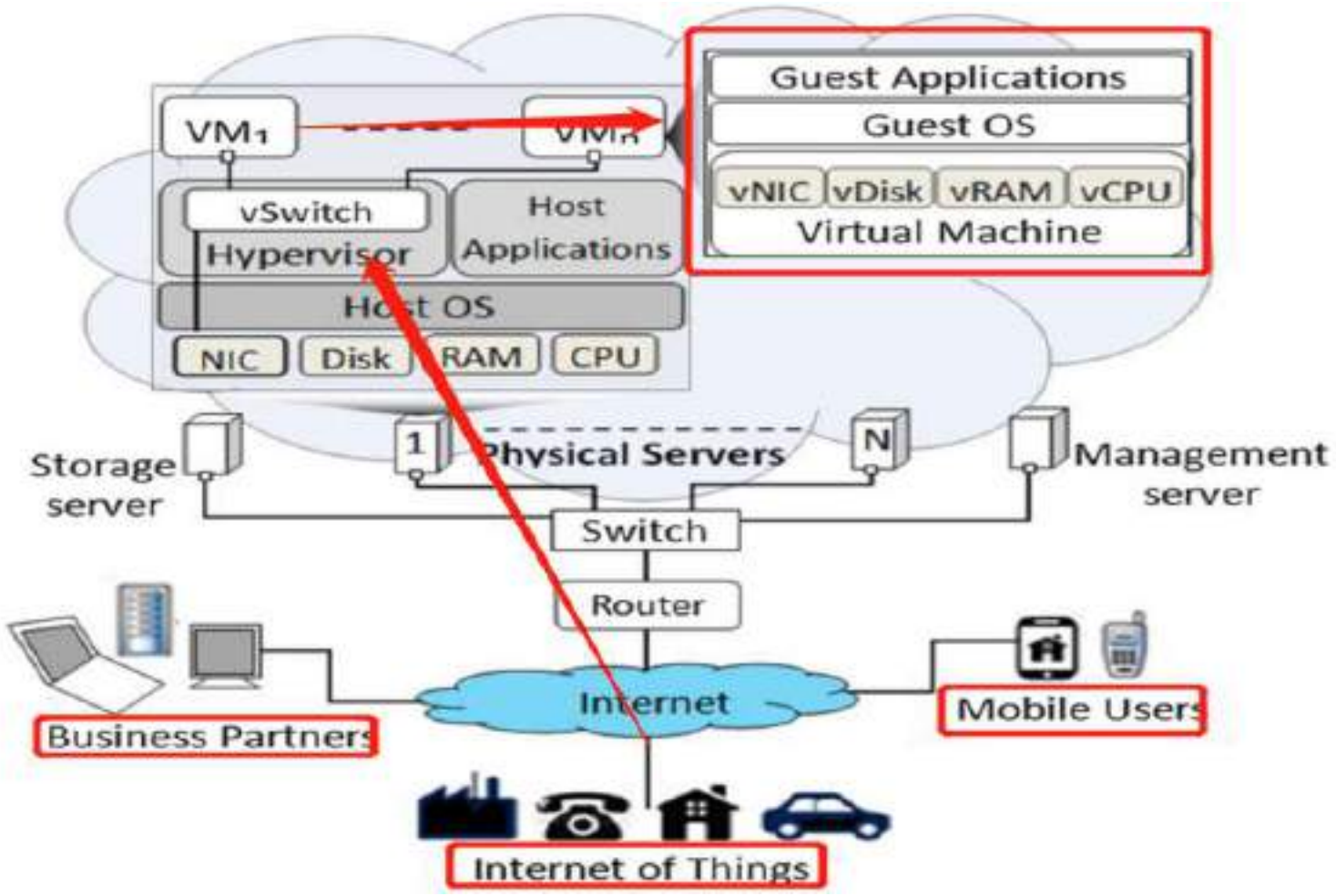
- 没有网络安全就没有国家安全
  - 如果无法有效保证虚拟化平台自身的安全性，将会给整个国家和社会带来极大的网络安全威胁
  - 云服务供应商发生的停机、故障等事件，不仅会导致信息资产的流失，还会直接造成巨大的经济损失
- 需要预防和保护虚拟化云平台免受各类攻击的侵害
  - 对虚拟化平台中的虚拟机进行异常检测
  - 及时发现虚拟机的异常行为，辅助平台管理员、虚拟机用户等及时分析异常原因并采取响应措施
  - 为保障虚拟机的安全与稳定运行提供有力支撑

基本概念

- 云计算 (Cloud Computing)
  - 美国国家标准和技术局
    - 云计算是一种模型，能支持便捷地按需通过网络访问一个可配置地共享计算资源池，包括网络、服务器、存储、应用程序、服务，共享池中的资源能够以最少的用户管理投入或最少的服务提供商互动实现快速供给和回收
  - 百度百科
    - 是基于互联网的相关服务的增加、使用和交付模式，通常涉及通过互联网来提供动态易扩展且经常是虚拟化的资源

基本概念

- 是分布式计算、并行计算、效用计算、网络存储、虚拟化、负载均衡、热备份冗余等传统计算机和网络技术发展融合的产物



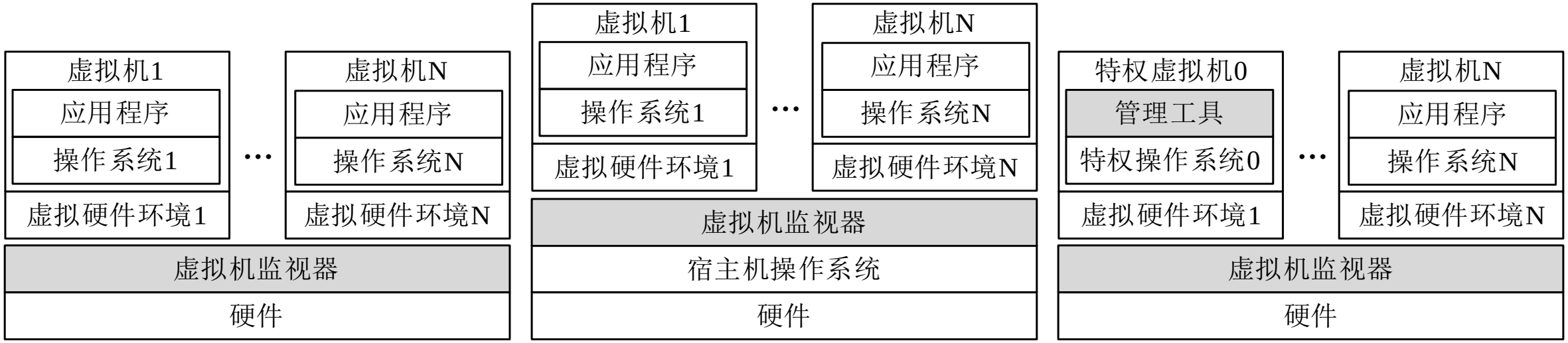
基本概念

- 虚拟化技术 (Virtualization)
  - 是指位于较低一层的软件模块，通过抽象出虚拟的软件或硬件接口，向上层软件模块提供它所期待的运行环境，使上层软件可以直接运行在虚拟环境中的方法
  - 虚拟化是云平台、在线虚拟机服务等的关键技术基础
  - 使得普通用户可以自由地按需使用虚拟设备，大幅提高了硬件资源的利用率
- 虚拟化的结构体系
  - 物理资源使用“宿主 (Host)”为定语，虚拟资源使用“客户 (Guest)”为定语
  - 物理计算机则称为宿主机 (Host Machine)，虚拟机则称为客户机 (Guest Machine)
  - 虚拟机监视器 (Virtual Machine Monitor, VMM)，虚拟机 (Virtual Machine)

  - 《虚拟化平台操作系统内核级恶意攻击行为及其检测技术-2019-05-19》

基本概念

- 计算机系统中各个层次的虚拟化
  - 比如硬件层、操作系统层、库函数层、应用程序层等
- 本文重点研究和讨论硬件抽象层上的虚拟化
  - 系统虚拟化是指将一台物理计算机系统虚拟化为多台虚拟计算机系统，每个虚拟计算机系统（即虚拟机）都拥有独立的虚拟机执行环境，包括虚拟CPU、虚拟内存、虚拟磁盘等抽象硬件设备，以及在其中运行的操作系统和应用系统等

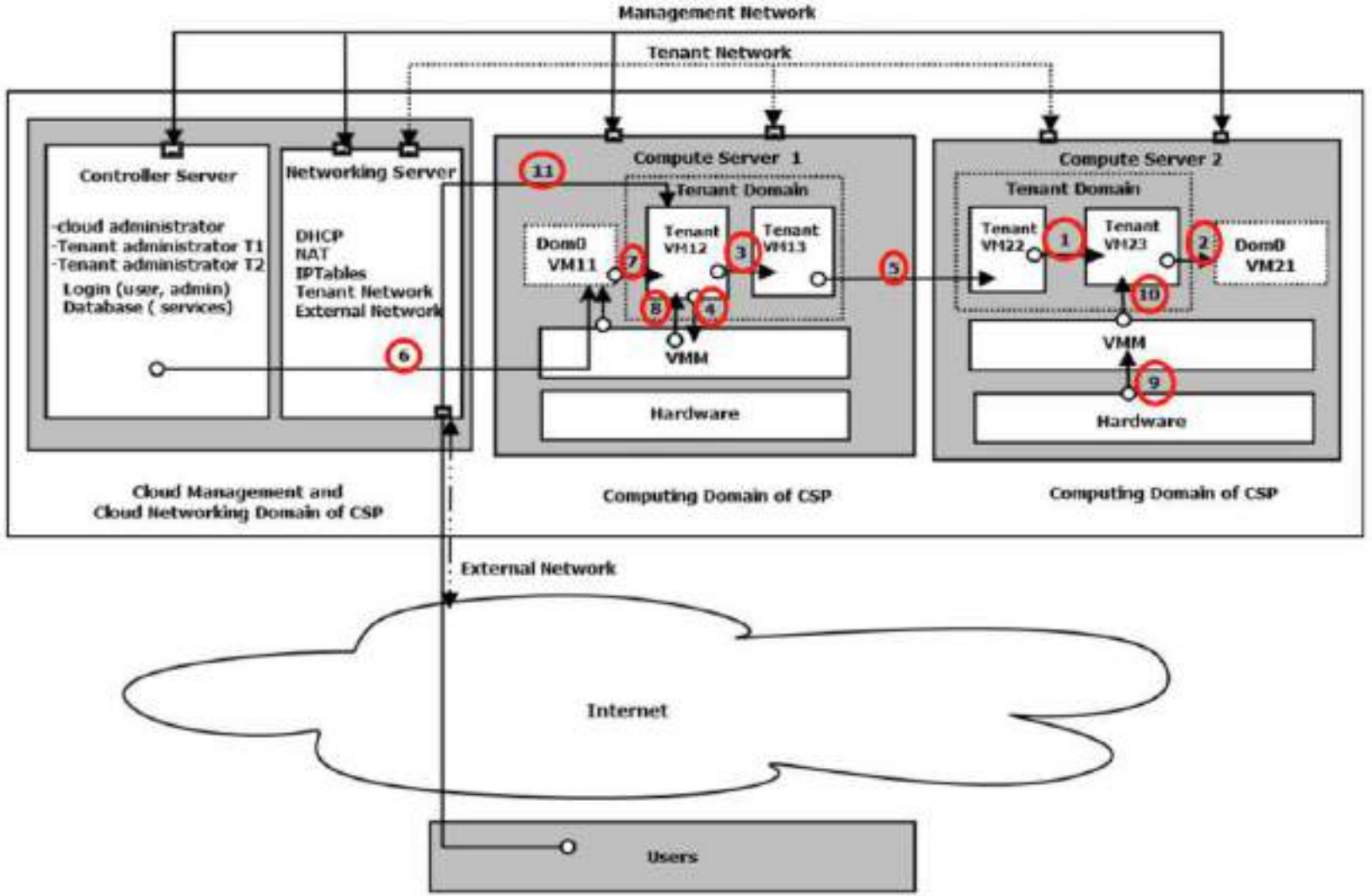


基本概念

- 云平台安全威胁模型
  - 三种类型服务器
    - 云控制服务器 **(Cloud Controller Server, CCS)**
    - 云计算服务器 **(Cloud Compute Server, CCoS)**
    - 云网络服务器 **(Cloud Networking Server, CNS)**
  - 三个网络
    - 租户网络、管理网络、外部网络
  - 四种用户角色
    - 云服务提供商 **(Cloud Service Provider, CSP)**
    - 云管理员 **(Cloud Administrator, CA)**
    - 租户管理员 **(Tenant Administrator, TA)**
    - 租户用户 **(Tenant User, TU)**

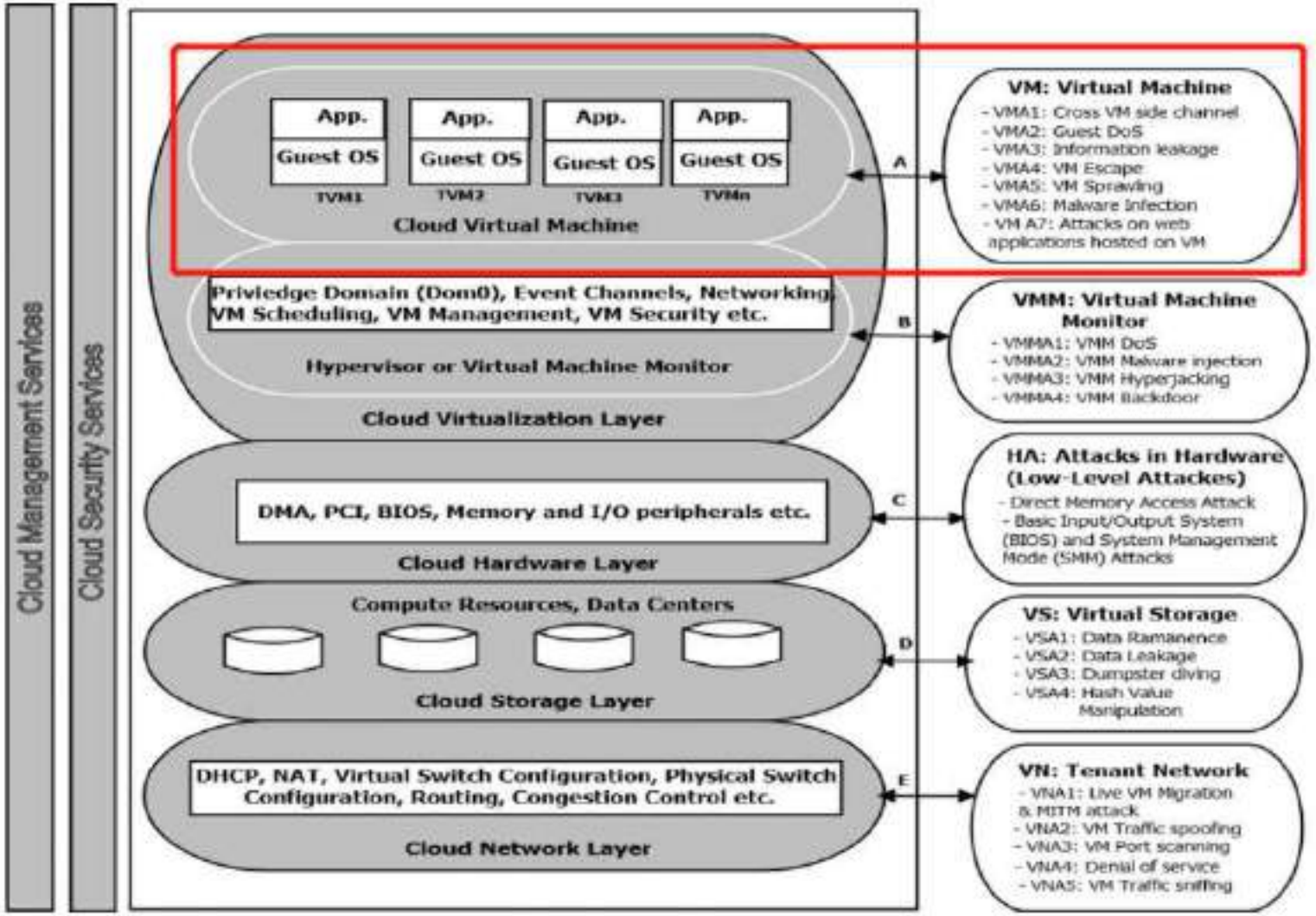
基本概念

- 云平台安全威胁模型
  - 1: 侧信道攻击
  - 2: 虚拟机逃逸
  - 3: Scanning
  - 4: Guest Dos
  - 5: 危险通信
  - 6/7: 内部威胁
  - 8: VMM受损
  - 9: 硬件越权
  - 10: VM受损
  - 11: 外部访问



基本概念

- 云平台安全威胁模型
  - **VMA1：侧信道**
  - **VMA2：Guest Dos**
  - **VMA3：信息泄露**
  - **VMA4：VM逃逸**
  - **VMA5：Sprawing**
  - **VMA6：恶意感染**
  - **VMA7：网络攻击**

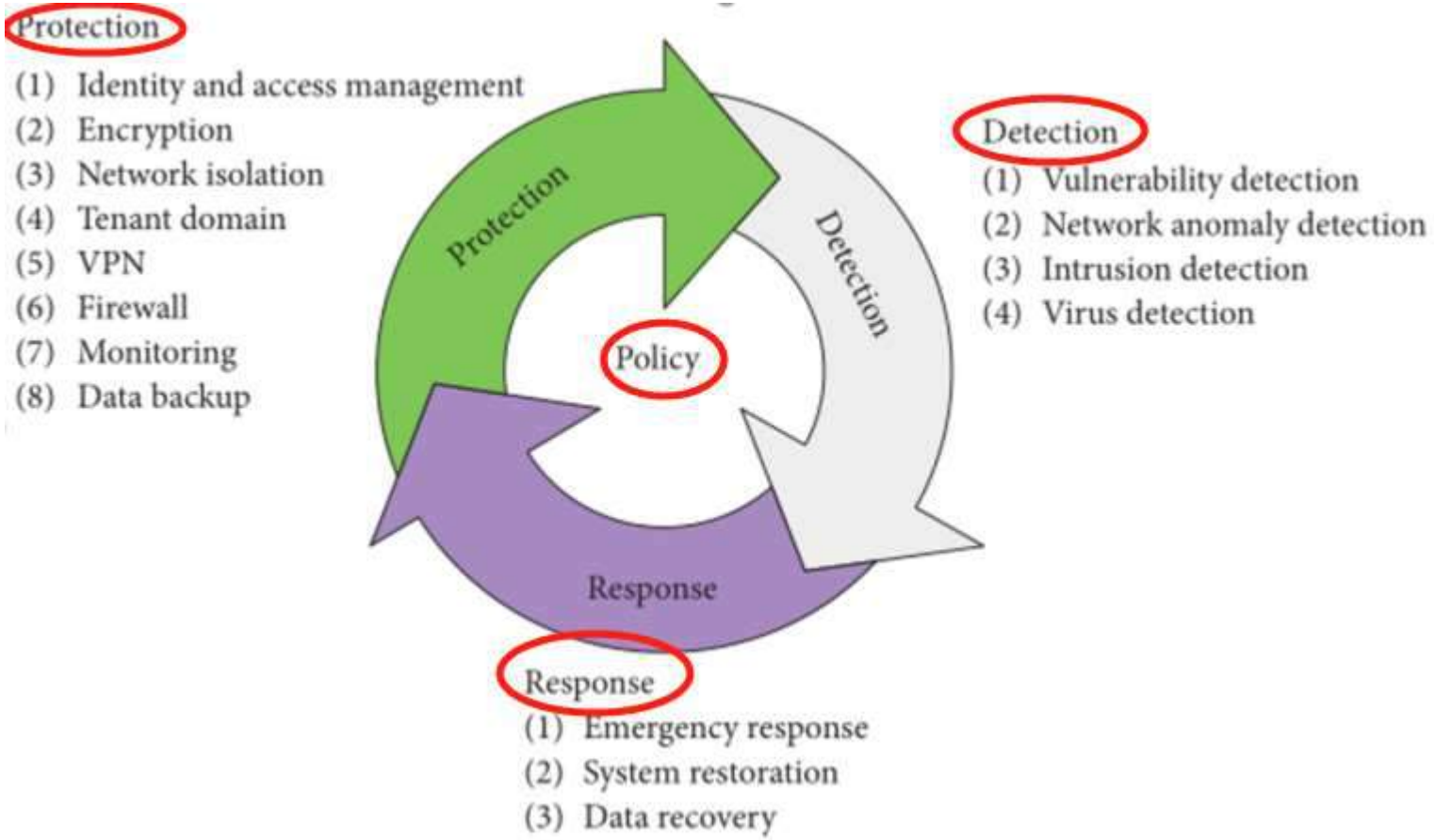


基本概念

- 策略-保护-检测-响应模型(Policy-Protection-Detection-Response，PPDR)
  - 策略
    - 模型的核心，所有的防护、检测和响应都是依据安全策略实施的，安全策略一般由总体安全策略和具体安全策略两部分组成
  - 保护
    - 保护是根据系统可能出现的安全问题而采取的预防措施，包括数据加密、身份认证、访问控制、虚拟专用网(VPN)技术、防火墙、安全扫描和数据备份等
  - 检测
    - 当攻击者穿透防护系统时，检测功能就发挥作用，与防护系统形成互补，检测是动态响应的依据
  - 响应
    - 系统一旦检测到人侵，响应系统就开始工作，进行事件处理，响应包括应急响应和恢复处理，恢复处理又包括系统恢复和信息恢复

基本概念

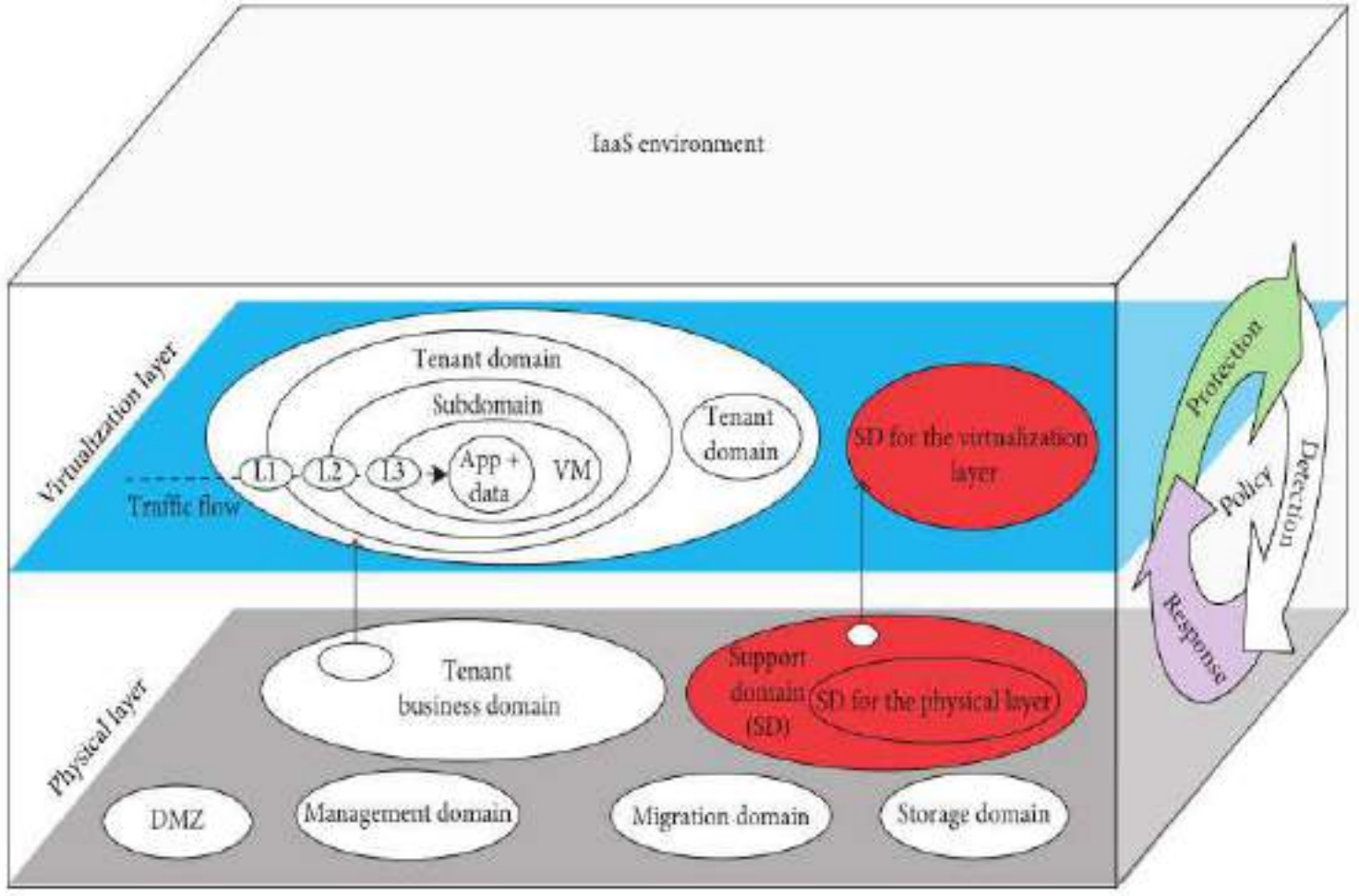
- 策略-保护-检测-响应模型(**Policy-Protection-Detection-Response，PPDR**)



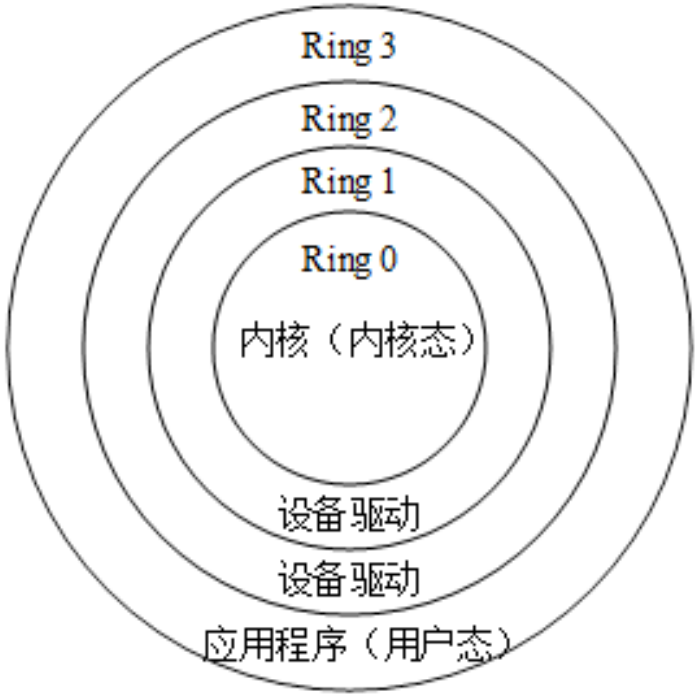
基本概念

- 策略-保护-检测-响应模型(Policy-Protection-Detection-Response，PPDR)

  - 安全策略控制和指导
  - 综合运用防护工具
  - 利用检测工具(人侵检测等)评估系统的安全状态
  - 通过适当响应将系统调整到“最安全”和“风险最低”状态
  - 保护、检测和响应组成完整、动态安全循环，在安全策略指导下保证信息系统安全
  - 系统、全面、深入、先进

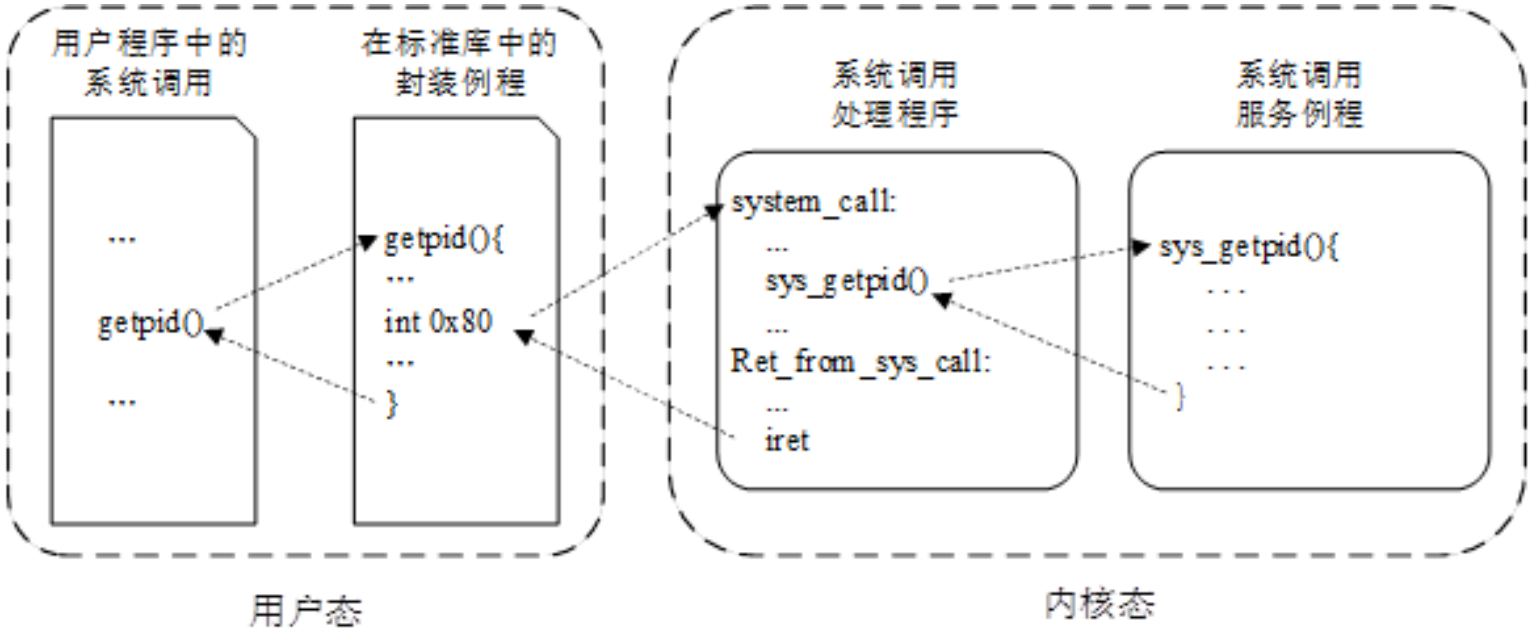


- 系统调用
  - 内核态和用户态
    - 有效管理和控制程序执行，并考虑操作系统安全
    - **CPU将特权级分4个级别：Ring0、Ring1、Ring2和Ring3**
    - 现代操作系统只使用**Ring0**（内核态）和**Ring3**（用户态）



基本概念

- 系统调用
  - 操作系统内核向用户态应用程序提供的接口
    - 运行在用户态的程序不能直接访问和修改操作系统内核数据结构
    - 用户态程序访问操作系统提供的核心功能，如修改内存、读写文件、创建进程等
    - 需要通过内核提供运行在**Ring0级别的接口**实现
    - 由内核模块提供的运行在**Ring0级别**的服务例程称为系统调用（**System Call**）



基本概念

- 系统调用
  - Linux操作系统
    - **64位Ubuntu18.04操作系统**
    - **/usr/include/asm/unistd_64.h**
    - 每个系统调用定义唯一的编号
    - **Linux不同版本之间可能有所不同**
    - **64位的Ubuntu18.04共定义332个**

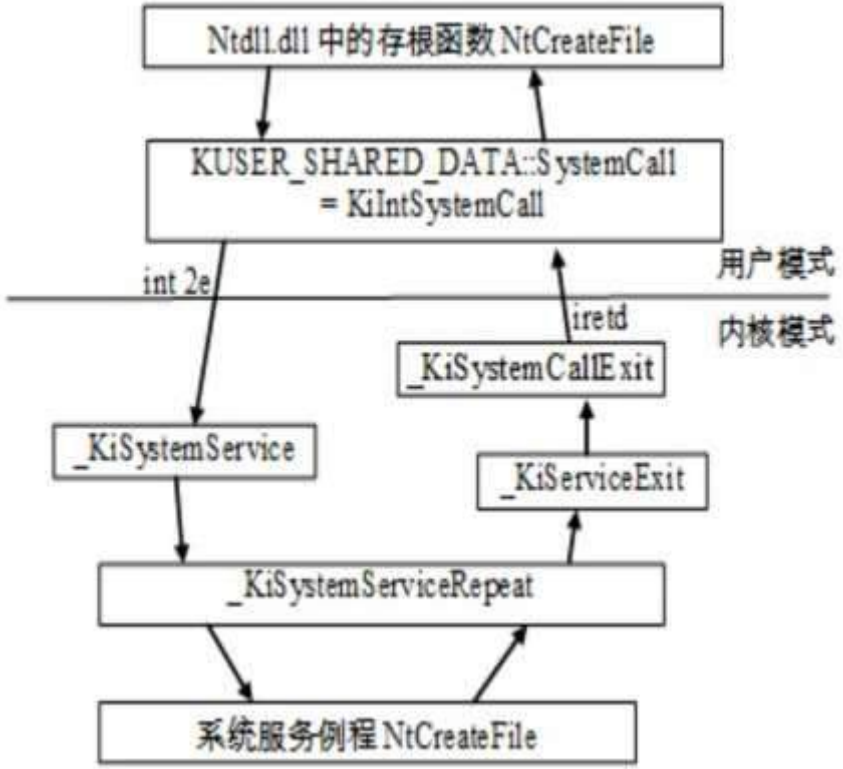
```
#define __NR_read 0
#define __NR_write 1
#define __NR_open 2
#define __NR_close 3
#define __NR_stat 4
#define __NR_fstat 5
...
...
#define __NR_pwritev2 328
#define __NR_pkey_mprotect 329
#define __NR_pkey_alloc 330
#define __NR_pkey_free 331
#define __NR_statx 332
```

基本概念

- 系统调用
  - Windwos操作系统
    - 与Linux操作系统类似
    - 中断法(int 2e)和快速法(sysenter)
    - 由动态链接库提供的API接口调用

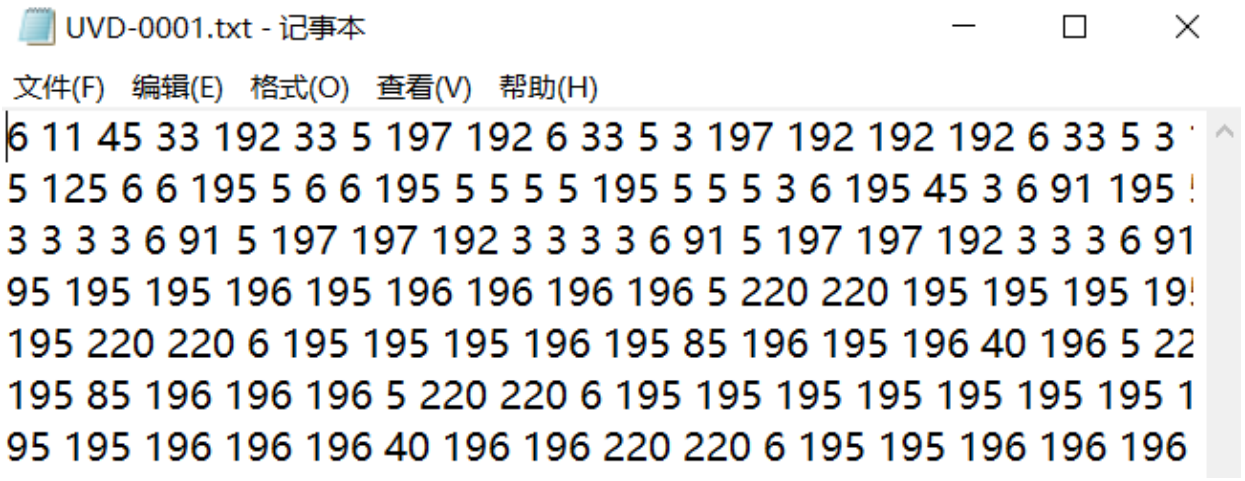
Windows X86-64 System Call Table (XP/2003/Vista/2008/7/2012/8/10)

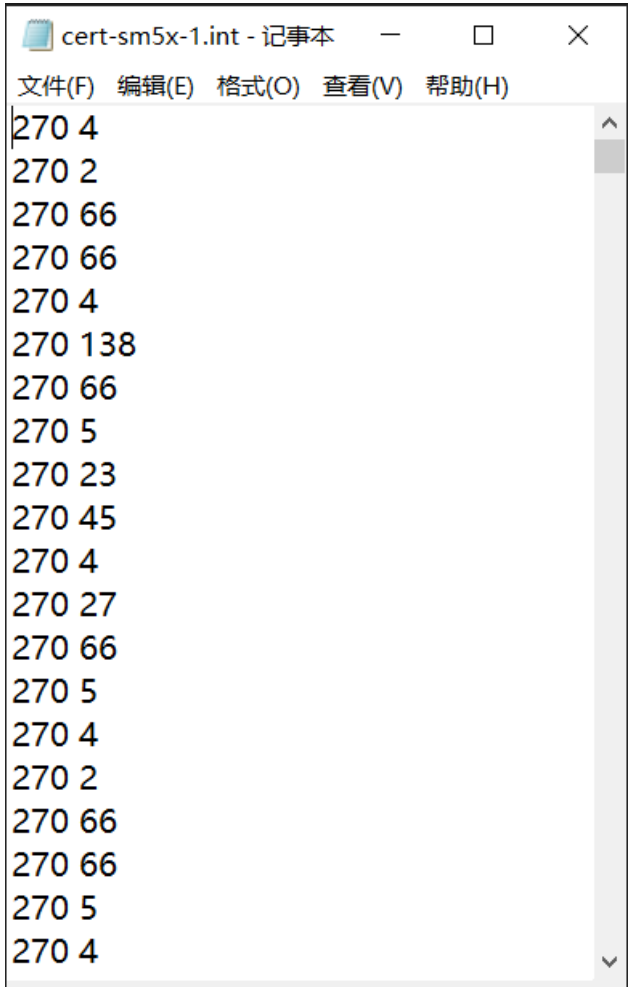
System Call Symbol	Windows 7 (hide)		Windows Server 2012 (hide)		Windows 8 (hide)		Windows 10 (hide)									
	SP0	SP1	SP0	R2	8	8.1	1507	1511	1607	1703	1709	1803	1809	1903	1909	2004
NtCreateEvent	0x0045	0x0045	0x0046	0x0047	0x0046	0x0047	0x0048	0x0048	0x0048	0x0048	0x0048	0x0048	0x0048	0x0048	0x0048	0x0048
NtCreateEventPair	0x0093	0x0093	0x009b	0x009c	0x009b	0x009c	0x009e	0x009f	0x00a0	0x00a3	0x00a4	0x00a5	0x00a5	0x00a6	0x00a6	0x00aa
NtCreateFile	0x0052	0x0052	0x0053	0x0054	0x0053	0x0054	0x0055	0x0055	0x0055	0x0055	0x0055	0x0055	0x0055	0x0055	0x0055	0x0055
NtCreateIRTimer			0x009c	0x009d	0x009c	0x009d	0x009f	0x00a0	0x00a1	0x00a4	0x00a5	0x00a6	0x00a6	0x00a7	0x00a7	0x00ab
NtCreateIoCompletion	0x0094	0x0094	0x009d	0x009e	0x009d	0x009e	0x00a0	0x00a1	0x00a2	0x00a5	0x00a6	0x00a7	0x00a7	0x00a8	0x00a8	0x00ac
NtCreateJobObject	0x0095	0x0095	0x009e	0x009f	0x009e	0x009f	0x00a1	0x00a2	0x00a3	0x00a6	0x00a7	0x00a8	0x00a8	0x00a9	0x00a9	0x00ad
NtCreateJobSet	0x0096	0x0096	0x009f	0x00a0	0x009f	0x00a0	0x00a2	0x00a3	0x00a4	0x00a7	0x00a8	0x00a9	0x00a9	0x00aa	0x00aa	0x00ae
NtCreateKey	0x001a	0x001a	0x001b	0x001c	0x001b	0x001c	0x001d	0x001d	0x001d	0x001d	0x001d	0x001d	0x001d	0x001d	0x001d	0x001d
NtCreateKeyTransacted	0x0097	0x0097	0x00a0	0x00a1	0x00a0	0x00a1	0x00a3	0x00a4	0x00a5	0x00a8	0x00a9	0x00aa	0x00aa	0x00ab	0x00ab	0x00af
NtCreateKeyedEvent	0x0098	0x0098	0x00a1	0x00a2	0x00a1	0x00a2	0x00a4	0x00a5	0x00a6	0x00a9	0x00aa	0x00ab	0x00ab	0x00ac	0x00ac	0x00b0
NtCreateLowBoxToken			0x00a2	0x00a3	0x00a2	0x00a3	0x00a5	0x00a6	0x00a7	0x00aa	0x00ab	0x00ac	0x00ac	0x00ad	0x00ad	0x00b1
NtCreateMailslotFile	0x0099	0x0099	0x00a3	0x00a4	0x00a3	0x00a4	0x00a6	0x00a7	0x00a8	0x00ab	0x00ac	0x00ad	0x00ad	0x00ae	0x00ae	0x00b2
NtCreateMutant	0x009a	0x009a	0x00a4	0x00a5	0x00a4	0x00a5	0x00a7	0x00a8	0x00a9	0x00ac	0x00ad	0x00ae	0x00ae	0x00af	0x00af	0x00b3



基本概念

- 系统调用
  - **ADFA数据集**
  - **UNM数据集**





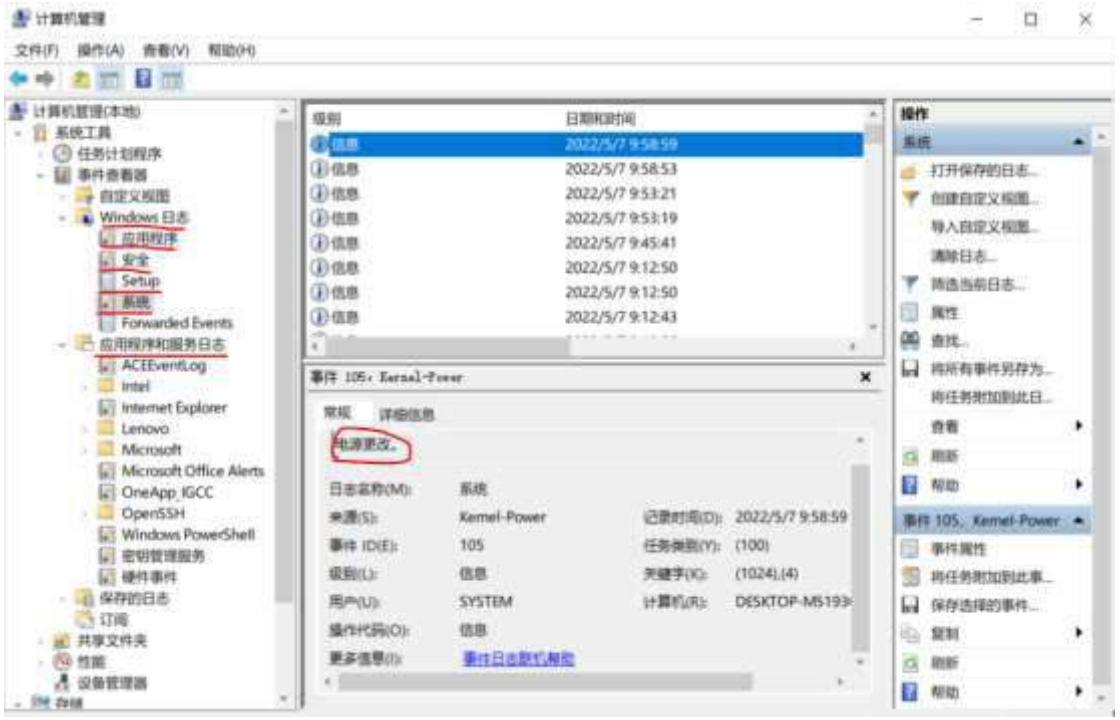
基本概念

- 系统日志
  - 系统日志是记录系统中硬件、软件和系统问题的信息
  - 监视系统中发生的事件
  - 检查错误发生的原因，或者寻找受到攻击时攻击者留下的痕迹
    - 系统日志帮助管理员/用户充分了解系统运行环境情况
    - 对于决定故障的根本原因或者缩小系统攻击范围来说是非常关键的
    - 了解故障或者袭击发生之前的所有事件
  - 系统日志包括系统日志、应用程序日志和安全日志

基本概念

- **Linux日志系统**
  - **Syslog、Syslog-ng**
- **Windwos日志系统**
  - 事件查看器

日志文件	说明
/var/log/cron	记录了系统定时任务相关的日志
/var/log/cups	记录打印信息的日志
/var/log/dmesg	记录了系统在开机时内核自检的信息，也可以使用dmesg命令直接查看内核自检信息
/var/log/mailog	记录邮件信息
/var/log/message	记录系统重要信息的日志。这个日志文件中会记录Linux系统的绝大多数重要信息，如果系统出现问题时，首先要检查的就应该是这个日志文件
/var/log/btmp	记录错误登录日志，这个文件是二进制文件，不能直接vi查看，而要使用lastb命令查看
/var/log/lastlog	记录系统中所有用户最后一次登录时间的日志，这个文件是二进制文件，不能直接vi，而要使用lastlog命令查看
/var/log/wtmp	永久记录所有用户的登录、注销信息，同时记录系统的启动、重启、关机事件。同样这个文件也是一个二进制文件，不能直接vi，而需要使用last命令来查看
/var/log/utmp	记录当前已经登录的用户信息，这个文件会随着用户的登录和注销不断变化，只记录当前登录用户的信息。同样这个文件不能直接vi，而要使用w,who,users等命令来查询
/var/log/secure	记录验证和授权方面的信息，只要涉及账号和密码的程序都会记录，比如SSH登录，su切换用户，sudo授权，甚至添加用户和修改用户密码都会记录在这个日志文件中



基本概念

- **Discounted Cumulative Gain，DCG**
  - 折损累计增益
  - 衡量搜索引擎返回结果列表质量优劣
    - 搜索引擎
    - 查询 **(Query)**
    - 返回结果列表：**list_1=[A,B,C,D,E]、 list_2=[D,A,E,C,B]**
  - 增益 **(Gain)**
    - 表示一个列表中所有$item$的相关性分数
    - $rel(i)$表示$item(i)$相关性得分：

$$Gain_i = rel(i)$$

基本概念

- **Discounted Cumulative Gain，DCG**
  - 累计增益 **(Cumulative Gain)**
    - 对给定数量k个$item$的Gain进行累加，只考虑相关性，不考虑$item$在返回列表中的位置

$$CG_k = \sum_{i=0}^{k} rel(i)$$

  - 假设**A,B,C,D,E**与查询**Query**之间对应的相关性为**0.5、0.9、0.3、0.6、0.1**，
    - 返回列表list_1=[A,B,C,D,E]的CG为：**0.5+0.9+0.3+0.6+0.1=2.4**
    - 返回列表list_2=[D,A,E,C,B]的CG为：**0.6+0.5+0.1+0.3+0.9=2.4**
  - 根据以上累计增益方式，无法对两个结果进行优劣比较

基本概念

- **Discounted Cumulative Gain，DCG**
  - 折损累计增益
    - 考虑排序顺序的因素，使得排名靠前的$item$增益更高，对排名靠后的$item$进行折损

$$DCG_k = \sum_{i=0}^{k} \frac{rel(i)}{log_2(i+1)}$$

  - 返回结果中的$item$顺序很重要，不同位置的贡献不同，一般来说，排在前面的$item$影响更大，排在后面的$item$影响较小
    - 例如一个返回的网页，肯定是排在前面的$item$会有更多人点击
  - DCG使排在前面的$item$增加其影响，排在后面的$item$减弱其影响
    - $\frac{1}{log_2(i+1)}$是一个非递增函数，i表示$item$的次序位置，其越大导致增益折扣越大

基本概念

- **Discounted Cumulative Gain，DCG**
  - 按照折损累计增益的方法，对于得到的两个返回结果，其DCG分数计算过程如下：
    - **list_1=[A,B,C,D,E]：**

i	rel(i)	log(i+1)	rel(i)/log(i+1)
1 = A	0.5	1	0.5
2 = B	0.9	1.59	0.57
3 = C	0.3	2	0.15
4 = D	0.6	2.32	0.26
5 = E	0.1	2.59	0.04

    - **list_1的 DCG_1= 0.5+0.57+0.15+0.26+0.04=1.52**

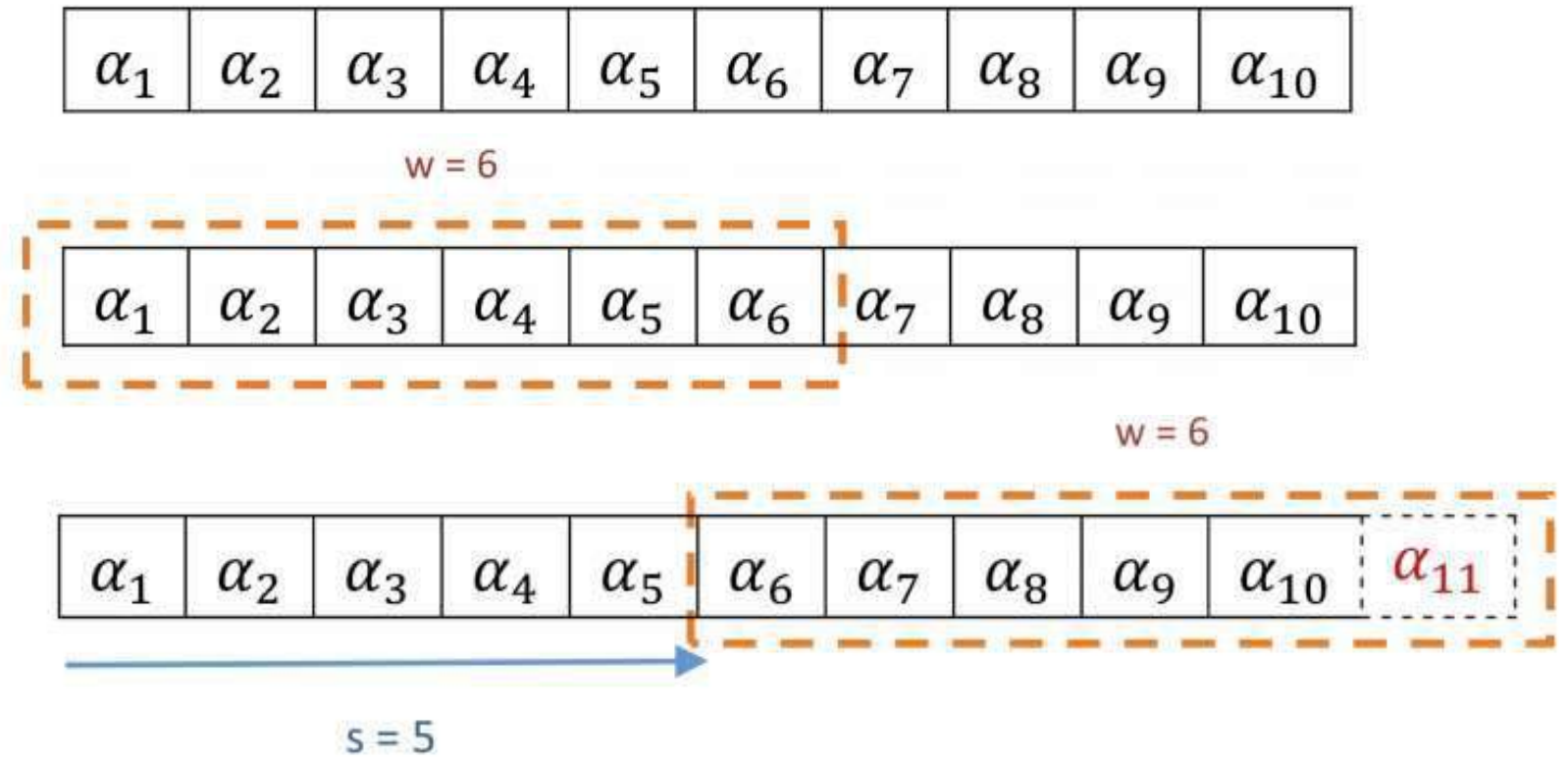
- **Discounted Cumulative Gain，DCG**
  - 按照折损累计增益的方法，对于得到的两个返回结果，其DCG分数计算过程如下：
    - **list_2=[D,A,E,C,B]：**

i	rel(i)	log(i+1)	rel(i)/log(i+1)
1 = D	0.6	1	0.6
2 = A	0.5	1.59	0.31
3 = E	0.1	2	0.05
4 = C	0.3	2.32	0.13
5 = B	0.9	2.59	0.35

    - **list_2的 DCG_2= 0.6+0.31+0.05+0.13+0.35=1.44**
  - **DCG_1 > DCG_2, 所以在这个例子里list_1优于list_2**

基本概念

- **N-gram方法**
  - 通过长度为**window_size (N)**的滑动窗口
  - 将窗口沿序列从头至尾方向逐渐移动**step_size**获得子序列



算法原理

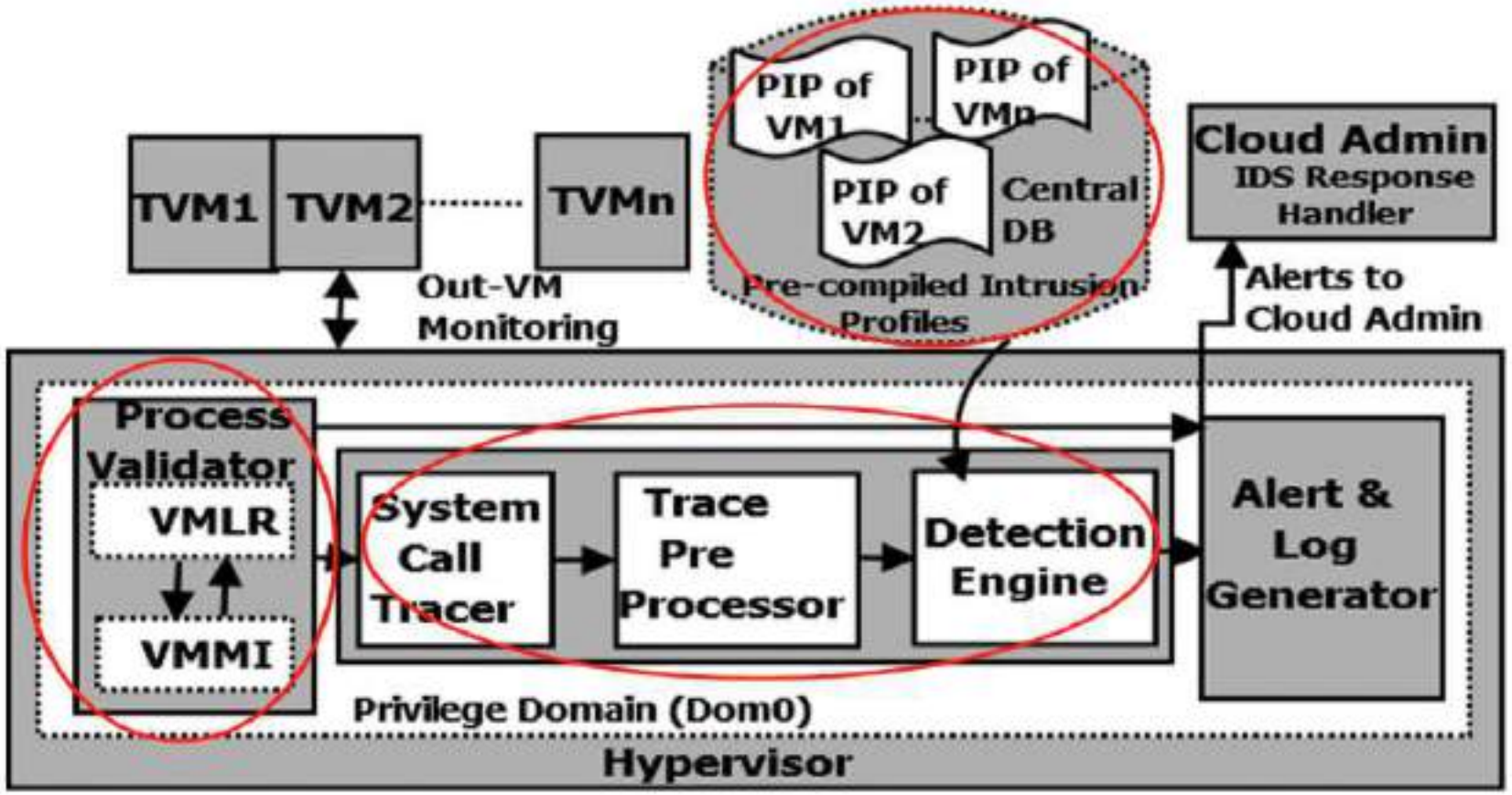
算法原理 VMGuard

T	检测云平台中针对虚拟机的入侵攻击
I	目标虚拟机内进程列表、目标虚拟机内应用程序的系统调用跟踪序列
P	1.1. 长期观察并获取正常稳定运行状态下虚拟机的进程列表 1.2. 分别获取目标虚拟机中非可信视图、可信视图的进程列表信息 1.3. 检查可信视图中安全关键进程，如不存在则判定**TVM**已遭受入侵攻击受损 1.4. 如果安全关键进程仍存在，将非可信视图进程列表与稳定状态进程列表、可信视图进程列表进行比较，如发现存在新进程、隐藏进程则判定**TVM**已遭受入侵攻击受损 2.1. 如通过第一阶段检查，则获取目标虚拟机内部所有进程的系统调用跟踪序列 2.2 使用**Bag of N-gram**方法提取特征向量，输入检测引擎进行识别 2.3 输出各目标进程所产生系统调用序列的类别，发现恶意进程则判定**TVM**已被入侵
O	目标虚拟机是/否已遭受入侵攻击受损

P	恶意软件篡改安全关键程序、隐藏自身等行为给检测带来困难
C	以云平台中虚拟机为对象，对其运行状态、内部进程异常行为进行检测
D	在进程级别和系统调用级别联合检测恶意程序
L	**2021 IEEE Trans SCI-2区**

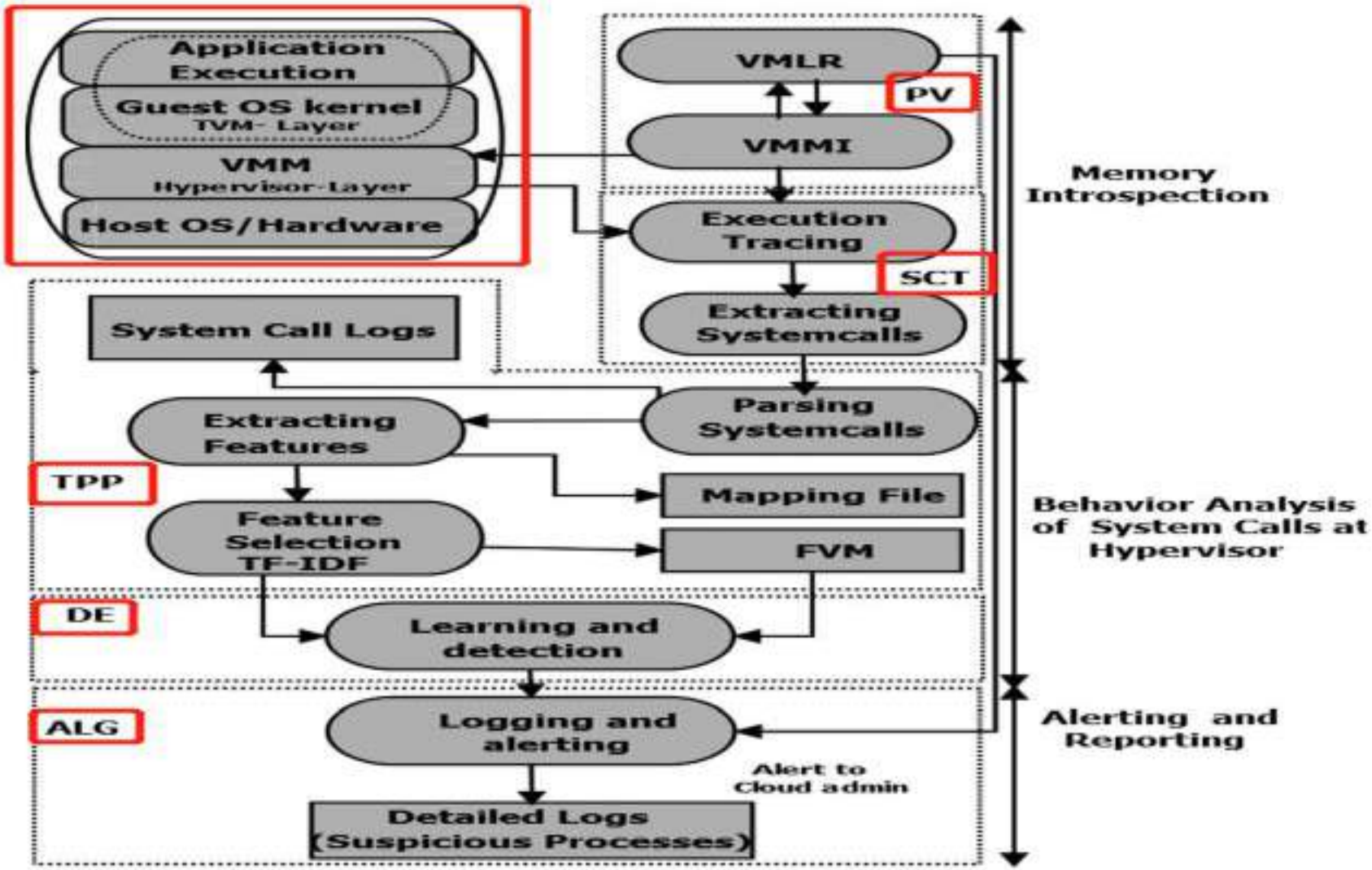
算法原理 VMGuard

- 总体框架
  - 主要组件：进程验证 (PV)、系统调用跟踪器（SCT）、跟踪预处理器（TPP）、检测引擎（DE）以及警报和日志生成器（ALG）



算法原理 VMGuard

- 工作流程



- 工作流程
  - 进程验证（**PV**）
    - 检测TVM中是否禁用了任何安全关键流程，如自动更新、自动扫描等
    - 检测TVM中是否有任何新进程或隐藏进程运行
  - 具体方法
    - 运行干净状态的虚拟机操作系统，**TVM**内使用**tasklist**或**ps**，**TVM**外使用**LibVMI API**接口从**VMM**获取进程列表，将此作为白名单，辅助验证虚拟机内进程
    - 主要针对恶意软件可以改变现有进程、调用新的恶意进程或禁用**TVM**中运行的安全关键进程，如自动更新或自动扫描操作
  - 子模块
    - **VMLR**用于维护来宾操作系统的详细信息
    - **VMMI**用于从来宾操作系统提取运行时进程级别的信息

- 工作流程
  - 系统调用跟踪（SCT)
    - 以进程执行的有序系统调用序列的形式捕获在TVM中运行的程序的执行跟踪
  - 具体方法
    - 在虚拟机监控程序中收集系统调用跟踪序列
    - 为每个受监控的TVM准备一个日志文件（Log_DB），以维护TVM中当前正在运行的程序的执行跟踪
    - 从早期阶段提取的系统调用将在行为分析阶段进行分析
  - 子模块
    - **Execution Tracing** 进程执行跟踪
    - **Extracting Systemcalls** 提取系统调用序列

- 工作流程
  - 跟踪预处理器（TPP)
    - 对收集的系统调用跟踪进行预处理，从跟踪序列中提取所需的特征，并将其转发给检测引擎进行进一步分析
  - 具体方法
    - **Bag of N-gram**方法

1. Process each trace i (present in Log_DB) into a feature vector, represented by $X_i = n_1, n_2, n_3, n_4, \ldots\ldots n_m$, as a set of n-grams, where m is the total number of n-grams obtained after processing the trace. Each n-gram is represented by a feature vector $n_j = < s_1 s_2 s_3 s_4 s_5 s_6 >$ where $1 <= j <= m$, a substring of a system call.
2. Each feature vector is converted into a numeric vector represented by $X'_i = < c_1, c_2, c_3, c_4, c_k >$ where each c value represents the total number of occurrences of each unique n-gram value in the trace and k is the total number of unique n-grams obtained.

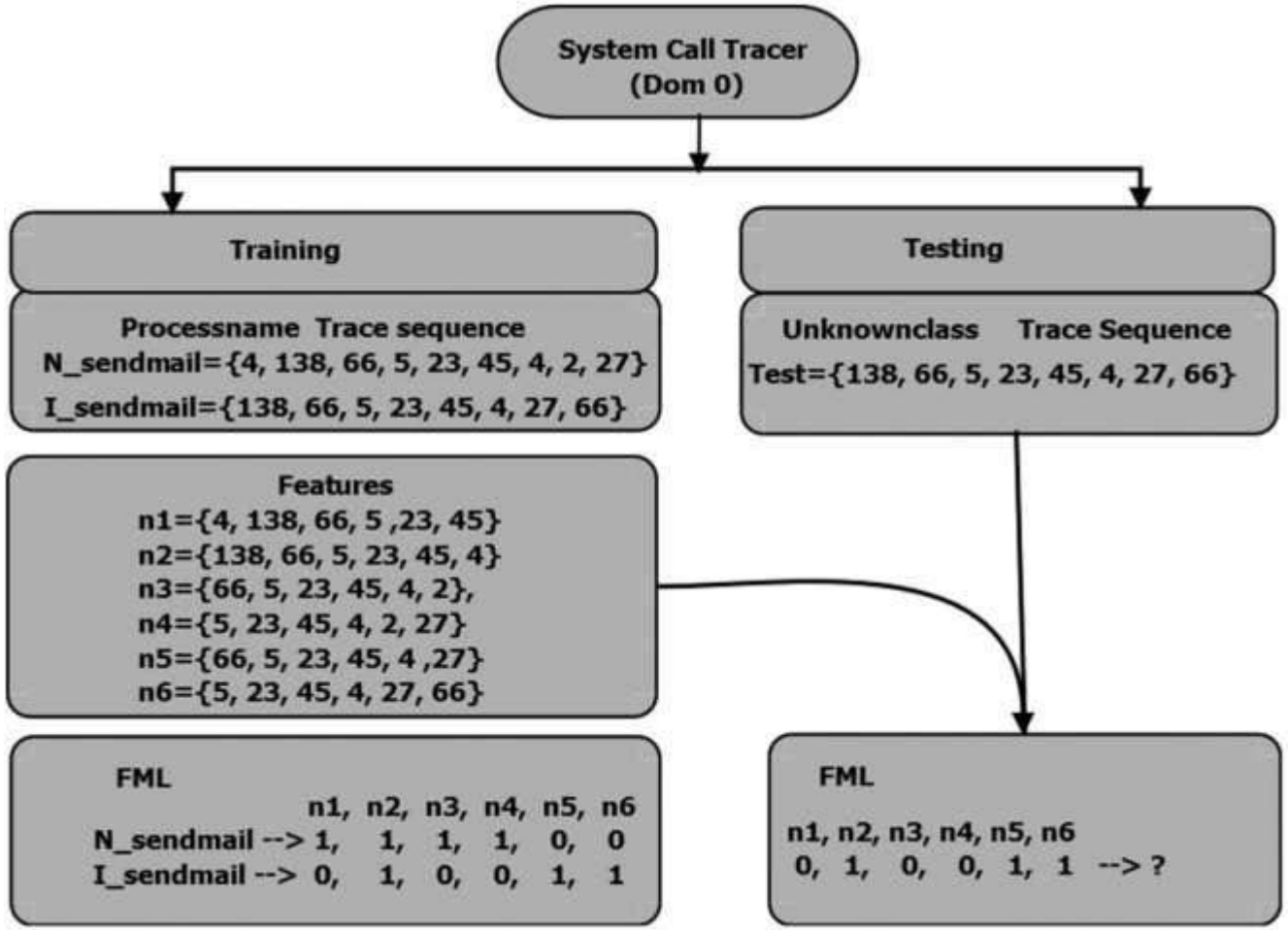
- 工作流程
  - 具体方法
    - 特征提取

Normal_sendmailprocess= {write→sethostid→sstk→open→ setuid→setgid→write→fork→ alarm}.
Intrusive_sendmailprocess= {sethostid→sstk→open→setuid→ setgid→write→ alarm→sstk}.

Sample of Unique n-Grams of Sendmail Process

Feature variable	N-grams representation	Actual n-grams
n1	[4, 138, 66, 5, 23, 45]	[write, sethostid, sstk, open, setuid, setgid]
n2	[138, 66, 5, 23, 45, 4]	[sethostid, sstk, open, setuid, setgid, write]
n3	[66, 5, 23, 45, 4, 2]	[sstk, open, setuid, setgid, write, fork]
n4	[5, 23, 45, 4, 2, 27]	[open, setuid, setgid, write, fork, alarm]
n5	[66, 5, 23, 45, 4, 27]	[sstk, open, setuid, setgid, write, alarm]
n6	[5, 23, 45, 4, 27, 66]	[open, setuid, setgid, write, alarm, sstk]

- 工作流程
  - 特征提取
    - 正常序列有n1、n2、n3和n4
    - n2也出现在侵入性序列
    - n5和n6只出现在入侵痕迹中
    - 可能是攻击模式
    - 对检测很重要
    - 每条序列中唯一**N-gram**的频次



- 工作流程
  - 特征选择
    - 为了提高泛化性能，降低分类器计算成本
    - 频次特性没有反应罕见性
    - 罕见性系统调用模式有利于在大型数据集上用于样本的快速类别区分
    - 罕见的系统调用序列通常代表系统的异常使用，对异常检测更加有用
    - 影响一个唯一N-gram特征子序列重要性的因素两个
      - 该特征子序列在目标序列中发生的频次（Term Frequency）
      - 该特征子序列在一系列跟踪序列中的罕见性（Inverse Document Frequency）

- 工作流程
  - 特征选择(**TF-IDF**)
    - **N-gram**特征子序列的频次定义
      - **N-gram**特征子序列在跟踪中出现的总次数（良性或恶意）

$$TF(\mathrm{ab}, X_i) = f(V_{ab}, X_i)$$

      - $f(V_{ab}, X_i)$是指类别为i（良性或者恶意）的系统调用序列X中ab的频次
      - **N-gram**特征子序列的反文档频率定义

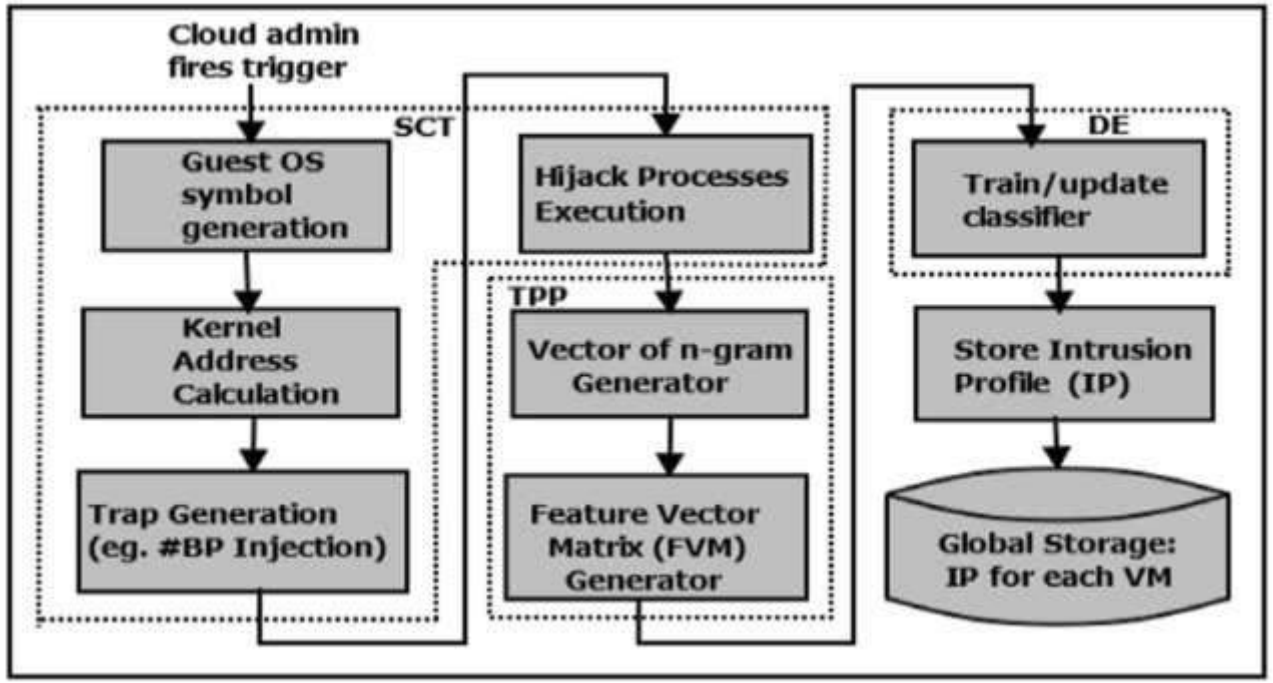
$$IDF(ab, C_i) = \log\left(\frac{C_i}{C_{V_{ab}}}\right)$$

      - C_i是指类i（良性或者恶意）的系统调用序列总数，$C_{V_{ab}}$表示包含ab的序列数量
      - **TF-IDF**是至以上二者的乘积

$$TF - IDF(\mathrm{ab}, C_i) = f(V_{ab}, X_i) * \log\left(\frac{C_i}{C_{V_{ab}}}\right)$$

算法原理 VMGuard

- 工作流程
  - 特征选择(TF-IDF)
    - 根据TF-IDF值的大小对备选特征N-gram子序列进行筛选作为最终的特征子序列
    - 注意，在选择特征子序列时，是将良性序列和恶意序列分开计算TF-IDF值得
  - 检测引擎、警报和日志生成



- 实验结果
  - 数据集
    - UNM公开数据集
      - 新墨西哥州立大学发布的UNIX单进程系统调用跟踪序列数据集
    - Dnew自建数据集
      - 运行良性进程、恶意进程，收集对应的系统调用跟踪序列

Details of University of New Mexico (UNM) Dataset

Process	Number of Intrusive traces considered for training	Number of Normal traces Considered for training	Total n-grams considered (after TF-IDF processing)
CERT synthetic sendmail	23	294	2,312
MIT live lpr	460	1,001	11,201
UNM synthetic lpr	1,001	9	14,384
UNM live lpr	1,001	1,231	16,959
UNM Live ps	26	24	866
UNM Live stide	105	645	7,321

- 实验结果
  - 随机森林、支持向量机（SVM）、朴素贝叶斯（NB）和集成SVM分类器结果

Detection Rate (%) of Different Classifiers for UNM Dataset

Data Set Name	Ensemble Classifier (Random Forest)	SVM	Naive Bayes	Ensemble classifier (boosting with SVM)
CERT_synthetic _sendmail	98.7382	97.4763	62.4606	98.7382
MIT_live_lpr	99.9316	98.13	97.3306	98.69
UNM_live_lpr	99.9552	98.6	98.2975	99.121
UNM_live_stide	99.6	97.3333	99.6	98.131
UNM_ps	94	84	90	90
UNM_synthetic _lpr	100	99.12	96.8317	99.12

- 实验结果
  - 随机森林分类器结果

Performance Results of Random Forest Technique for UNM dataset

Process Name	Correctly Classified Instances	Incorrectly Classified Instances	TPR (%)	FPR (%)	FNR (%)
CERT_synthetic _sendmail	313	4	98.74	4.1	1.28
MIT_live_lpr	1,460	1	99.93	1	0.1
UNM_live_lpr	2,231	1	99.96	1	0
UNM_live_stide	747	3	99.6	2.5	0.4
UNM_ps	47	3	94	6.2	6
UNM_synthetic _lpr	1,010	0	100	0	0

算法原理 实验结果

- 实验结果
  - 对比实验结果

Comparison of Proposed Technique with Existing Work

Parameter	VMGuard [VMG]	BoS] [6]	ISCS [5]	Xenini-IDS [16]	ShadowContext [18]
Technique	VMI with Dynamic Analysis	Dynamic Analysis	Dynamic Analysis	VMI with Dynamic Analysis	VMI with Dynamic Analysis
Motive	Attack Detection	Attack Detection	Attack Detection	Attack Detection	Attack Prevention
VMI Technique	Break Point Injection	NA	NA	Interrupt Handling	System Call Redirection
Placement of IDS	Dom0	VM	VM	Dom0 and VMM	Dom0 and VMM
Feature Extraction	Vector of n-grams (Numeric)	Bag of System Calls (Numeric)	Key- Value Pair (String)	System Call Sequence (String)	System Call (String)
Ordering of System Call	Considered	Not Considered	Not Considered	Considered	Considered
Attack Resistance	High	Low	Low	Medium	Medium
Feature Selection	TF-IDF	NA	NA	NA	NA
False Positives	Low	High	High	High	NA
Storage Requirement	Medium	Medium	High	High	Low
Machine Learning	Random Forrest	NA	NA	NA	NA
Adaptability	More	Lesser than VMG	Lesser than VMG	Lesser than VMG	Lesser than VMG
Process Validation	Considered	Not Considered	Not Considered	Not Considered	Not Considered
Hidden Process Detection	Considered	Not Considered	Not Considered	Not Considered	Not Considered
Levels of Security Check	Two	One	One	One	One
Robustness of System	High	Low	Low	Medium	Medium
Detection Rate	94-100% (UNM) 93-99% (Dnew)	Nil (UNM), 100% (self generated)	4.7-100% (UNM)	100% (UNM sendmail)	NA
Hypervisor dependability	dependent	Any	Any	dependant	dependant

- 优势：
  - 为安全即服务的实际应用做了探索
  - 本文提出的方法对于CSP和Tenant都具有可借鉴意义
  - 使用了两个级别两个阶段联合检测的整体框架和思路
  - 所提出BoNG结合TF-IDF方法对恶意系统调用序列具有更高检测性能

- 局限性：
  - 依赖于Hypervisor虚拟机自省的功能，具有单点故障风险
  - 无法检测未知攻击和零日攻击等

算法原理

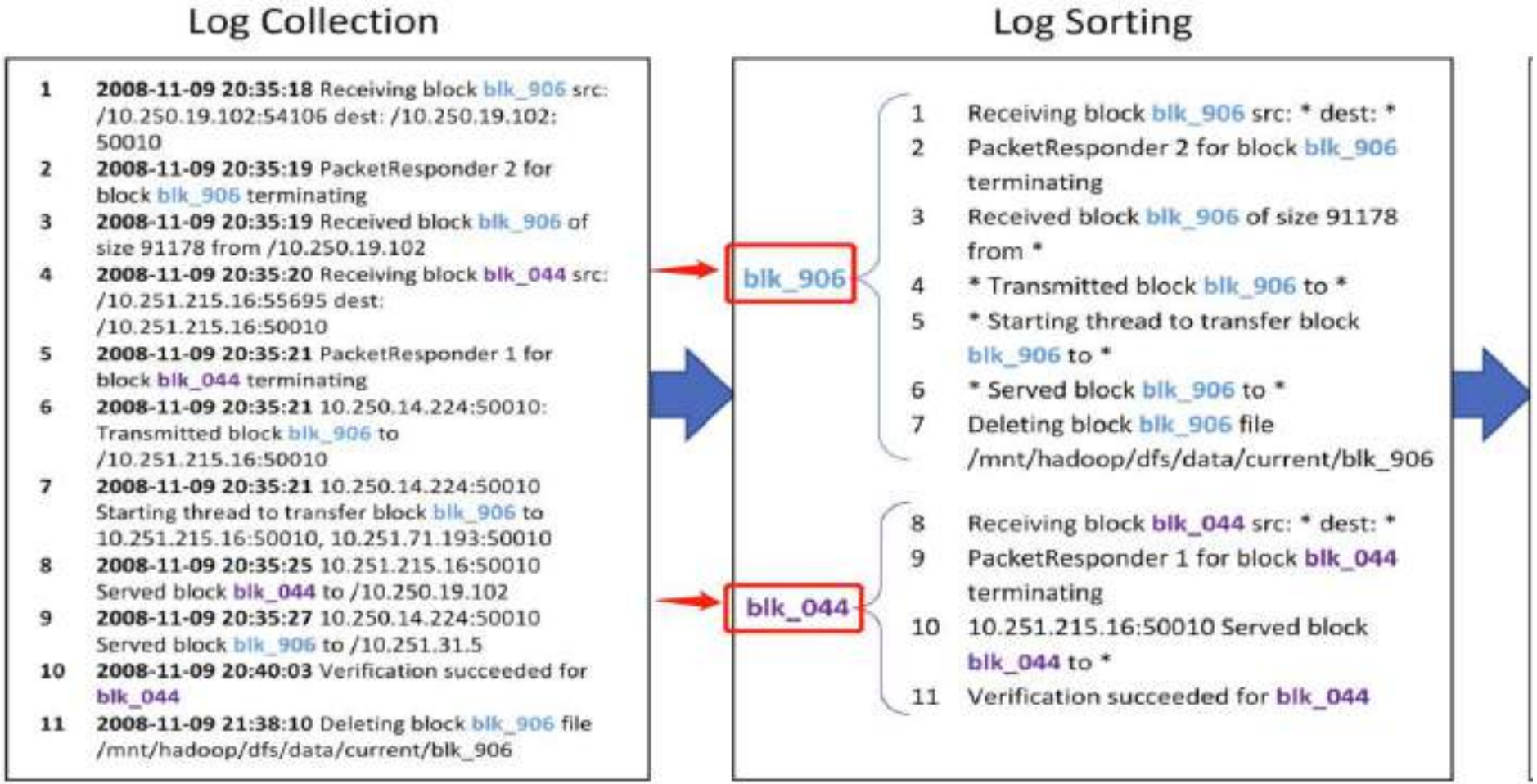
算法原理

T	检测系统运行日志的异常
I	应用系统运行日志数据
P	1. 日志收集 2. 日志分割，按照标识符对日志文本进行分组 3. 日志解析，获取日志模板、日志模板序列 4. 异常检测，确定候选日志模板，判定是否异常
O	待检测日志数据是/否存在异常

P	**Top-n候选日志模板确定方法易导致误报和漏报发生**
C	可正常解析日志数据
D	如何选择合适的候选日志模板集合（日志键集合）
L	**2019 通信顶会GlobeCOM**

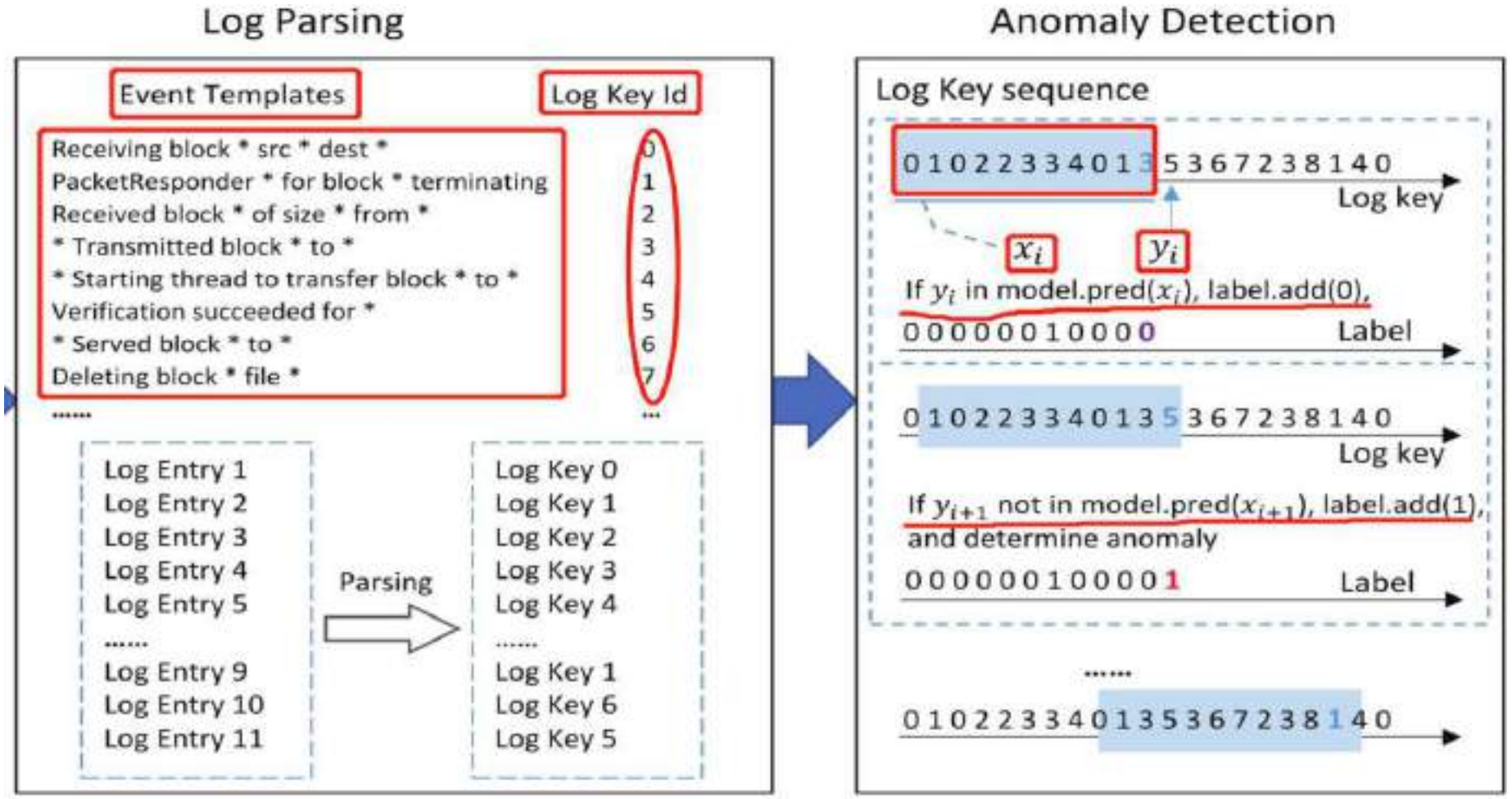
算法原理

- 总体框架
  - 日志收集、日志排序（日志分组）、日志解析、异常检测



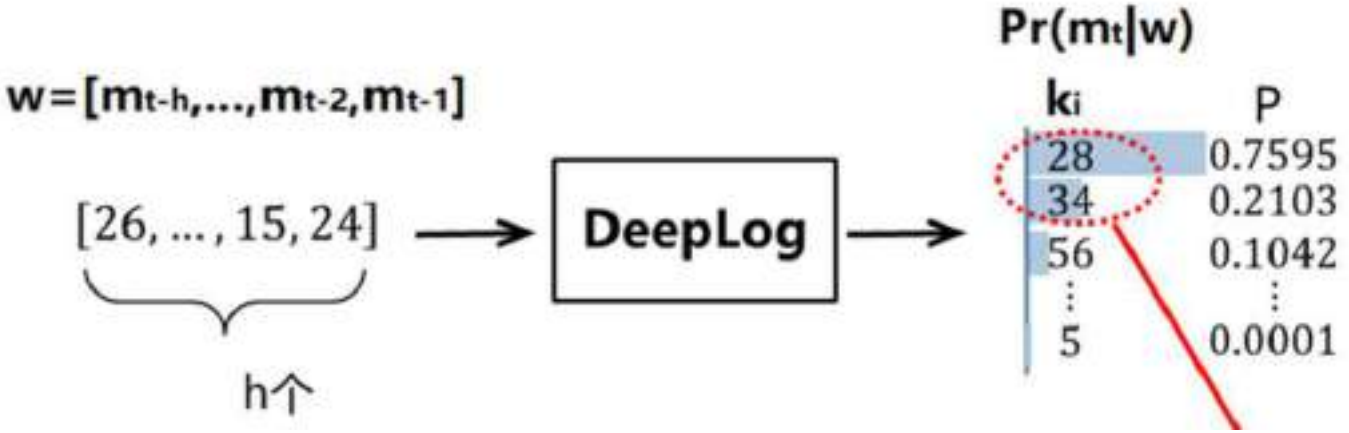
算法原理

- 总体框架
  - 日志收集、日志排序、日志解析、异常检测



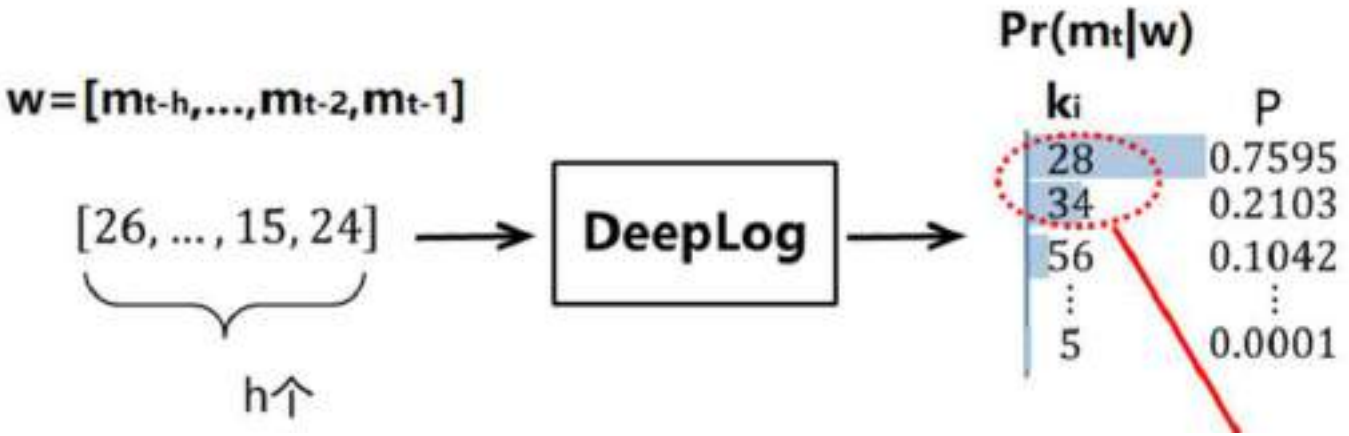
算法原理

- 实验框架
  - 训练阶段
    - 使用正常系统执行日志序列训练**LSTM**模型
  - 检测阶段
    - 已训练模型输出所有日志键上的预测概率分布
    - **DeepLog**等，将已训练模型预测出的具有前N个概率值的日志键中包含当前输入日志键序列的下一个日志键的情况视为正常
    - 误报：将潜在良好候选值排除（良判恶）或接受某些不适合候选值（恶判良）



算法原理

- 实验框架
  - 检测阶段
    - 如果已训练模型计算出某些类型的日志键在某个特定输入序列上的条件概率非常大，则可以认为此时的条件分支序列路径数很小，此时应选择较少的候选日志
    - 另一方面，如果条件概率分布被计算为平均值，那么此时条件分支序列路径的数量应该很大，此时应该选择更多的候选日志



算法原理

- 实验框架
  - 检测阶段
    - 使用DCG算法确定候选日志键

B. Discounted Cumulative Gain (DCG)

For an input sequence, the model(i.e., ranker) returns n ordered log keys. Suppose the log entries are originally ordered $\{1, 2, 3, ..., n\}$. The model will output a permutation mapping $\pi : \{1, 2, 3, ..., n\} \rightarrow \{1, 2, 3, ..., n\}$ [12].

The dcg score is computed from the conditional probability of the n log keys as

$$dcg = \sum_{i=1}^{n} c_{[i]}(2^{y_i} - 1), \quad (1)$$

where $[i]$ is the originally order, and $y_i \in [0, 1]$ is the conditional probability of the ith log key in the original (pre-ranked) order. $y_i = 1$ corresponds to a "perfect" relevance and $y_i = 0$ corresponds to a "impossible" relevance [12].

In the definition of DCG, $c_{[i]}$ which is a non-increasing function of i, is typically set as:

$$c_{[i]} = \frac{1}{\log(1 + i)}, i \in \{1, 2, 3, ..., n\} \quad (2)$$

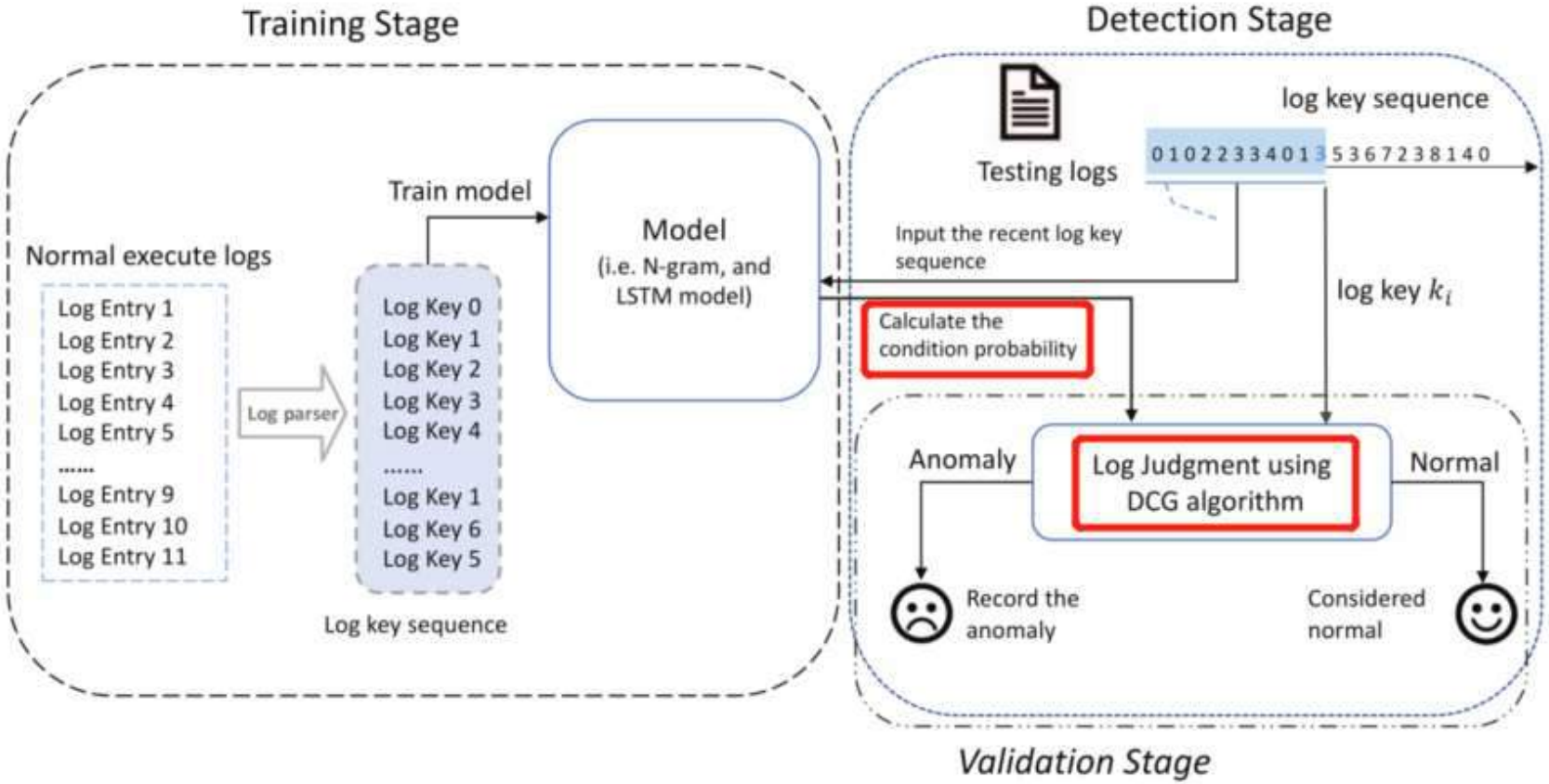
Algorithm 1 The DCG algorithm

Input: real log key list $true$, prediction for the probability of each log key appear $pred$, threshold dcg_score.
Output: The array of prediction truth value $pred\_label$.

create arrays $pred\_label[0...N-1]$
$pred\_index \leftarrow$ descending argsort $pred$
for $i = 0 \rightarrow N - 1$ **do**
  $cur\_dcg \leftarrow 0$
  create arrays $pred\_list[0...\|K\|-1]$
  for $k = 0 \rightarrow \|K\| - 1$ **do**
    $cur\_dcg = cur\_dcg + \frac{2^{pred_{i,pred\_index_{i,k}}} - 1}{\log_2(2+k)}$
    $pred\_list_k \leftarrow pred\_index_{i,k}$
    if $cur\_dcg \geq dcg\_score$ **then**
      break
    end if
  end for
  $pred\_label_i \leftarrow 0$ if $true_i \in pred\_list$, and $pred\_label_i \leftarrow 1$ otherwise
end for
return $pred\_label$

算法原理

- 实验框架
  - 检测阶段
    - 使用DCG算法确定候选日志键



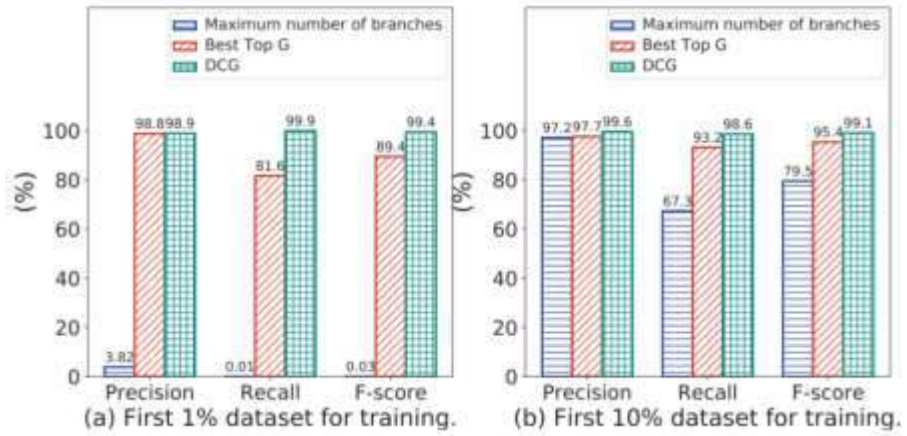
- 实验结果
  - 数据集
    - **HDFS公开数据集**
    - 用于基于日志的异常检测公共基准数据集。包含来自200多个亚马逊EC2节点的日志消息，可以根据block_id按会话窗口进行日志分组，已由Hadoop领域专家标记

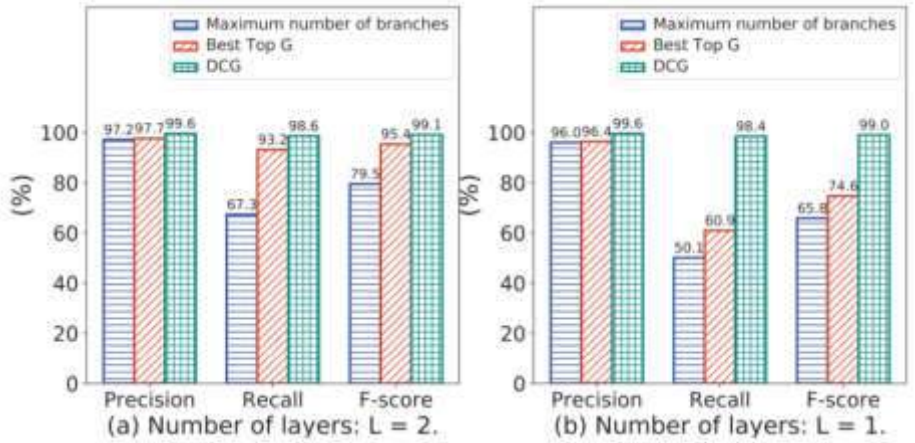
SET UP OF LOG DATA SETS(UNIT: SESSION)

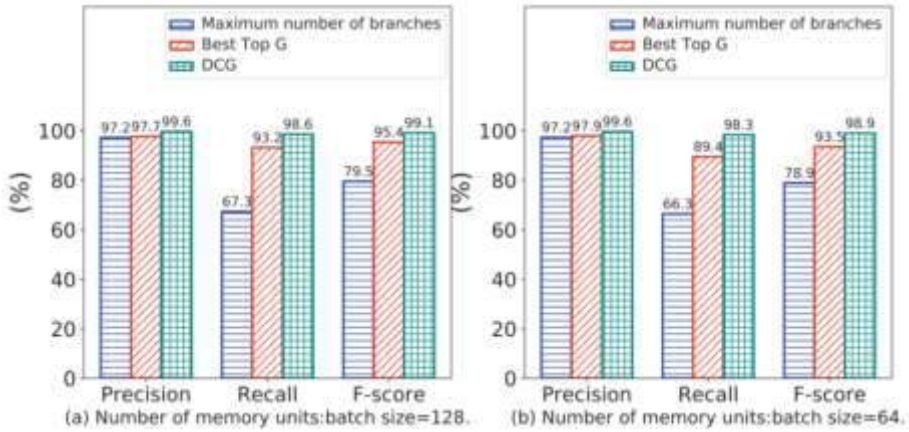
Log dataset	Number of sessions		
	training dataset	validation dataset	testing dataset
HDFS	54,029 normal; 0 abnormal	54,330 normal; 5764 abnormal	449,852 normal; 51,074 abnormal

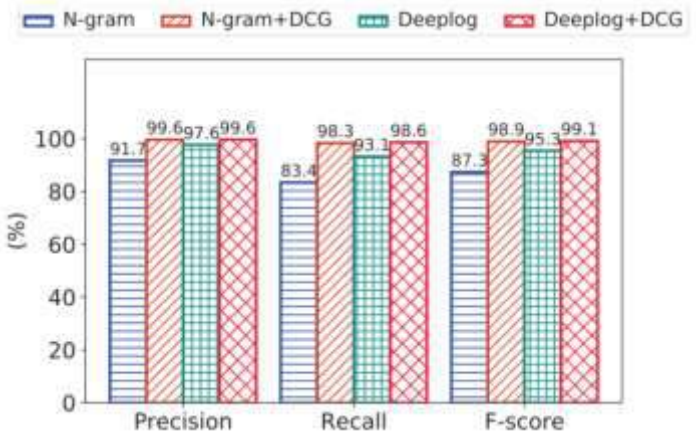
算法原理 实验结果

- 实验结果
  - 调参实验与对比实验









- 实验结果
  - 调参实验
    - 实验发现所提结合DCG算法的检测方法性能对于不同的值是相当稳定的
    - 也就是说，所提方法对这些参数值的任何一个或组合的调整都不太敏感
  - 对比实验

RESULT OF ANOMALY DETECTION PERFORMANCE AMONG DIFFERENT ALGORITHMS.

	FP	FN	Precision	Recall	F-score	Improvement
N-gram	33958	74676	91.7%	83.4%	87.3%	–
N-gram + DCG	1776	7648	99.6%	98.3%	98.9%	11.6%
Deeplog	10298	31040	97.6%	93.1%	95.3%	–
Deeplog + DCG	1781	6298	99.6%	98.6%	99.1%	3.8%

- 优势：
  - 使用DCG算法来确定日志判断的候选日志集合（截止值），而不是Top-N等固定截止值，避免排除潜在合适候选日志，或接受错误候选日志，提升检测准确率
  - 引入的DCG算法实用性和可解释性强

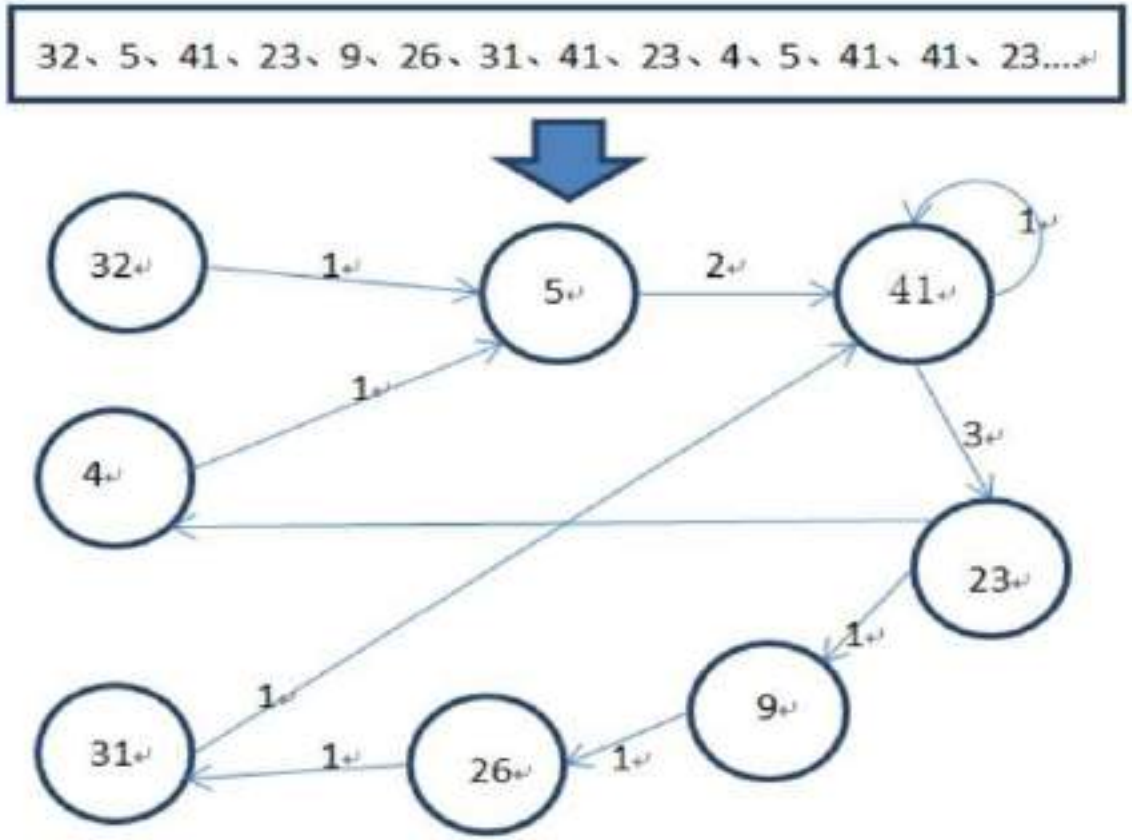
- 局限性：
  - DCG阈值确定仍需要额外开销和训练，方法对该值具有强敏感性

应用总结

算法原理

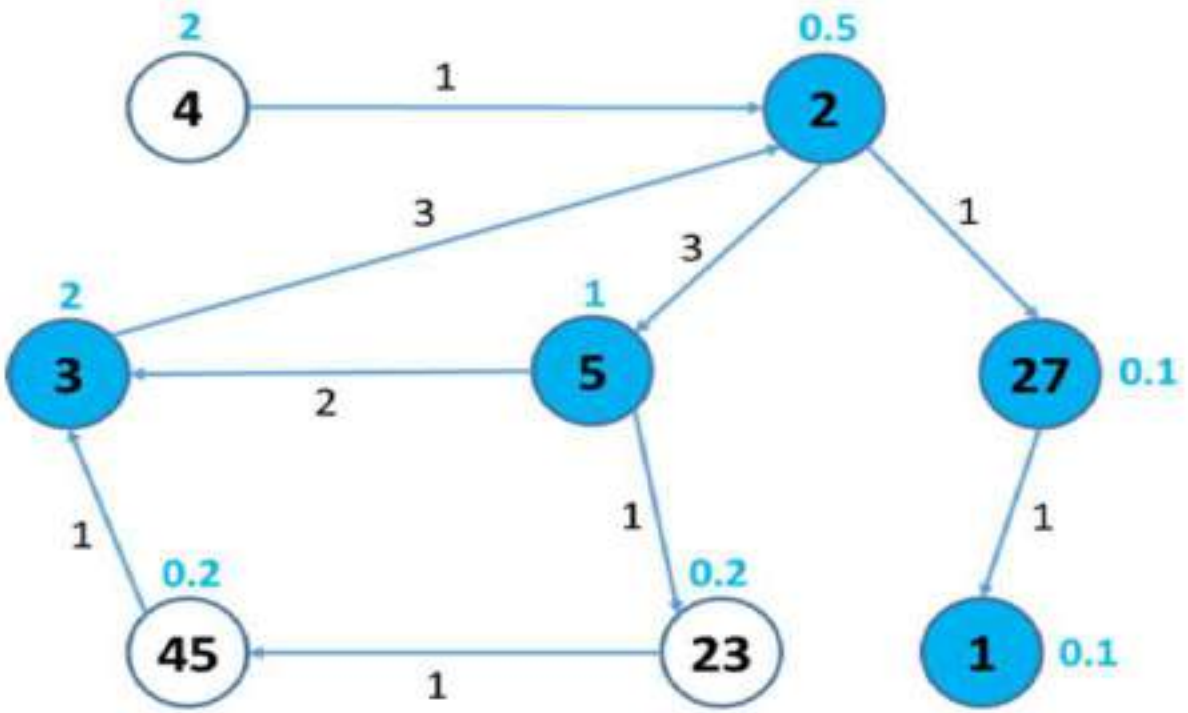
算法原理

- 序列数据转换为Graph
  - 将系统调用序列转换为有向的频次权重图
  - **An Anomaly Intrusion Detection Method Based on PageRank Algorithm**



算法原理

- 序列数据转换为Graph
  - 再增加节点信息，集成系统调用图**(Integrated System Calls Graph)**
  - 该节点的值通常是应用程序完全执行时调用该系统调用的频率或百分比
  - **Intrusion Detection System Based on Integrated System Calls Graph and Neural Networks**

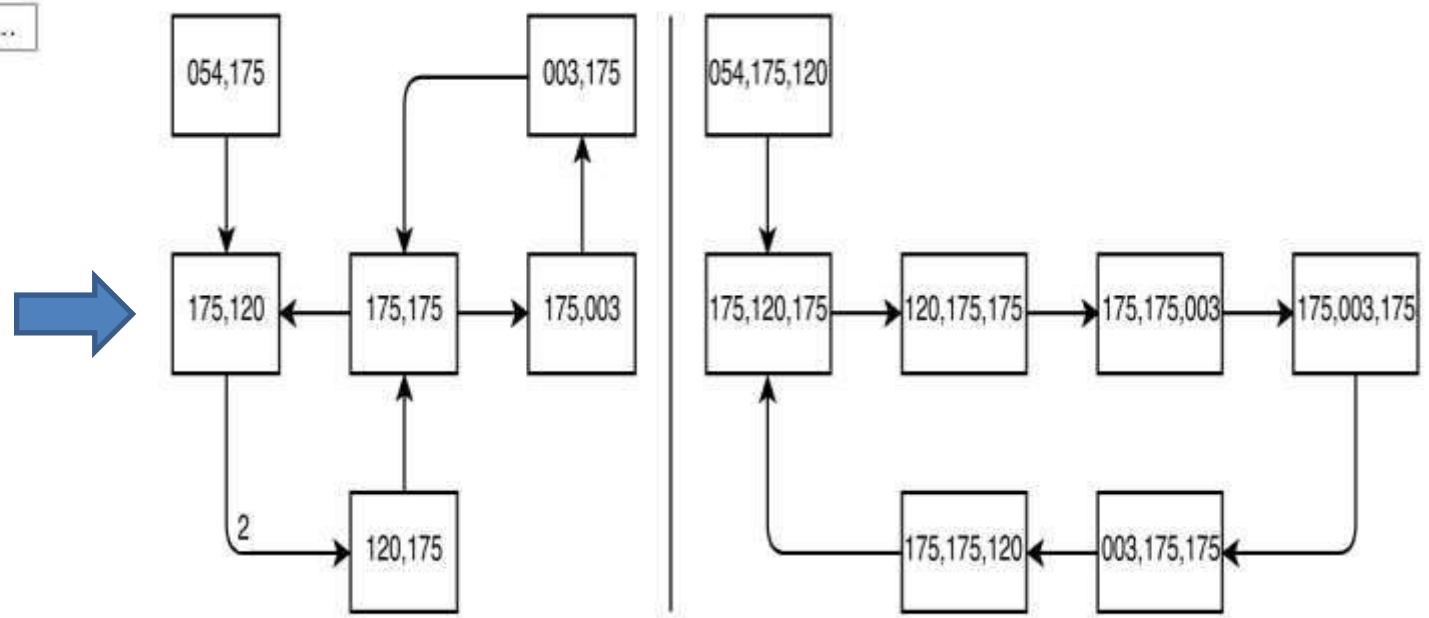


算法原理

- 序列数据转换为Graph
  - N-gram序列图，将N-gram子序列抽象为节点，权重为频次
  - **Intrusion Detection on System Call Graphs**

S:	054	175	120	175	175	003	175	175	120	175	...

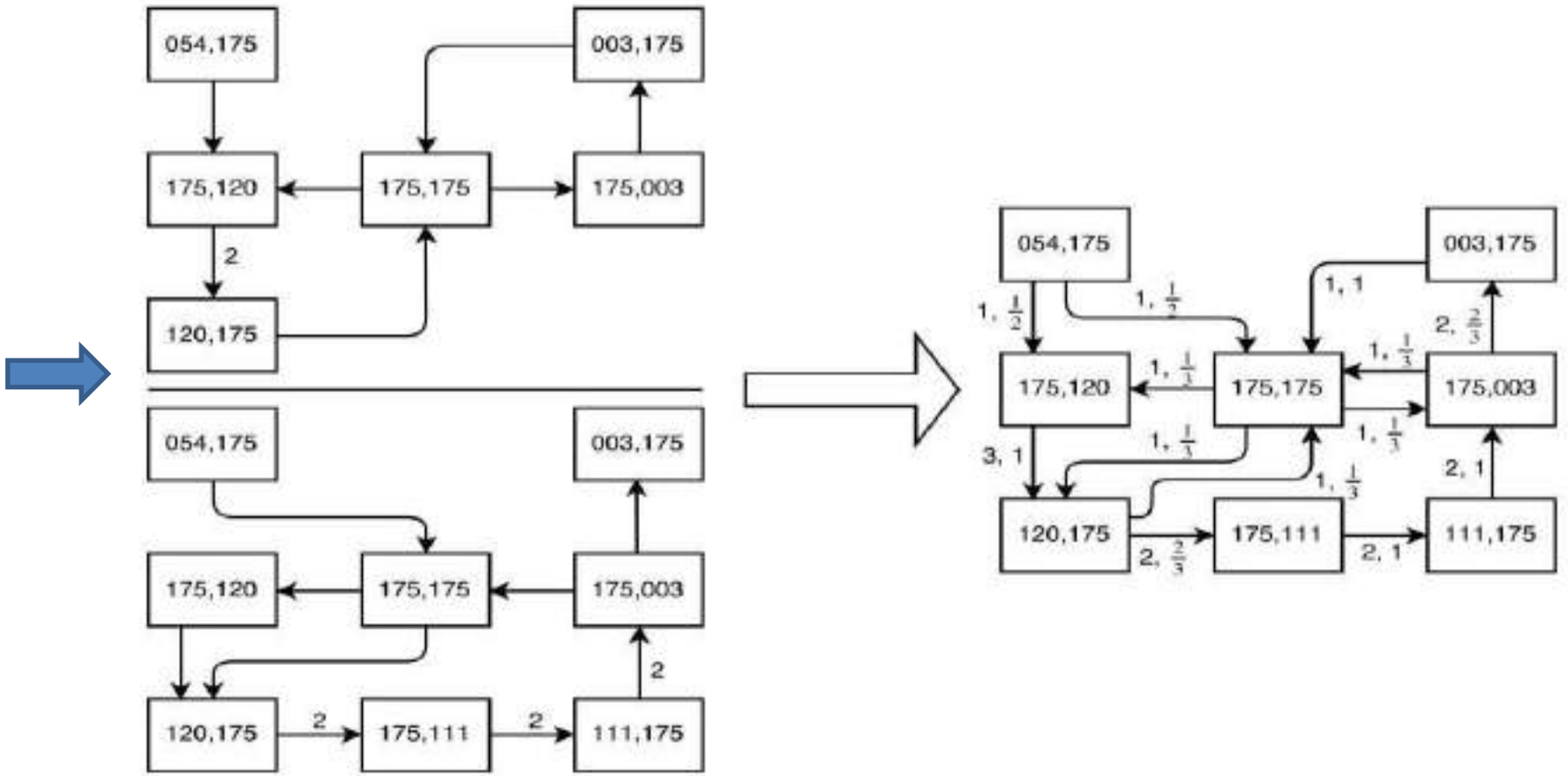
step 1	[**054**	**175**]	120	175	175	003	...
step 2	054	[**175**	**120**]	175	175	003	...
step 3	054	175	[**120**	**175**]	175	003	...



算法原理

- 序列数据转换为Graph
  - N-gram概率图，转移概率$p(a,b)$，过渡频率$f(a,b)$

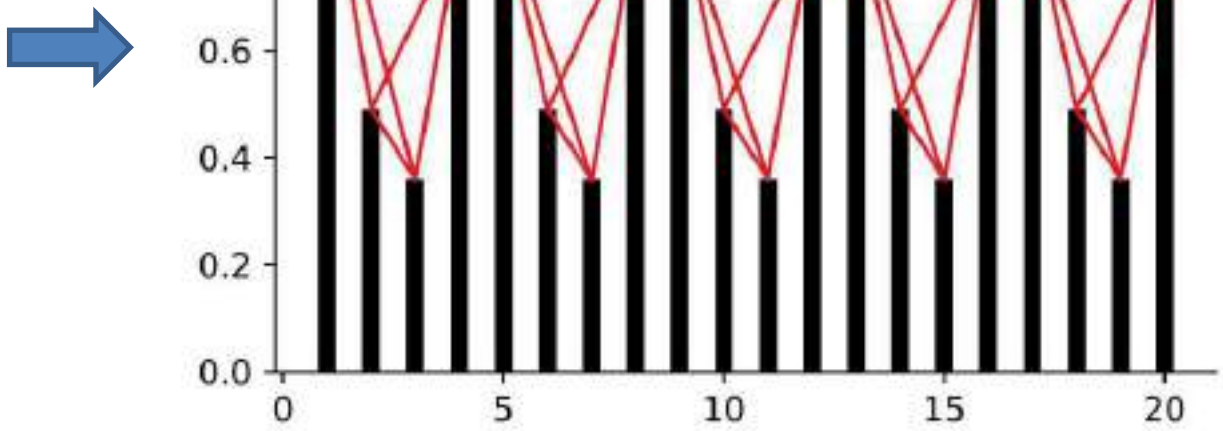
$$p(a,b) = \frac{f(a,b)}{\sum_{a_i \in A} f(a,a_i)}$$



算法原理

- 序列数据转换为Graph
  - 时间序列转换为能见度图（Visibility Graph），更关注全局特征
  - **Extracting Statistical Graph Features for Accurate and Efficient Time Series Classification**

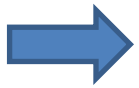
Definition 2.3 (Visibility graph). Given a time series $T = (v_1, \ldots, v_n)$, its VG representation $G = (V, E)$ has n vertices: $V = (1, \ldots, n)$. An edge $e = (i, j) \in E$ iff $\forall k$ such that $i < k < j$ $(1 \leq i, j \leq n)$ inequality $v_k < v_j + (v_i - v_j)\frac{j-k}{j-i}$ is satisfied.

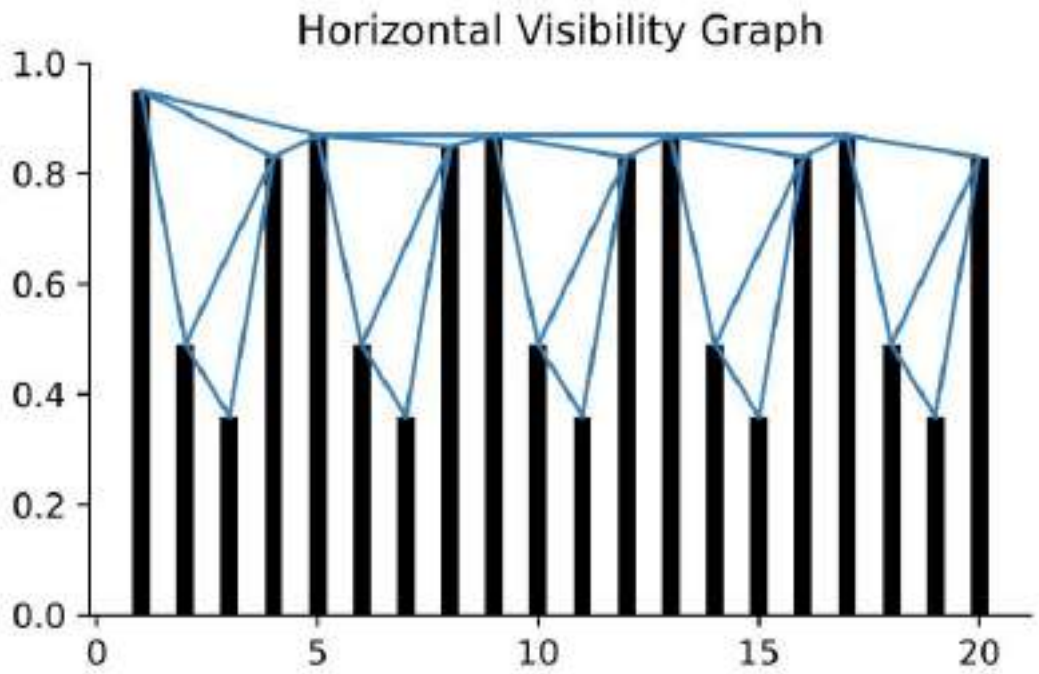


算法原理

- 序列数据转换为Graph
  - 时间序列转换为水平能见度图（**Horizontal Visibility Graph**），更关注局部特征
  - 均有仿射不变性，满足缩放不变性
  - 不适用于非平稳时间序列，单调增/减
  - 表征时间序列结构特性，也可定义加权

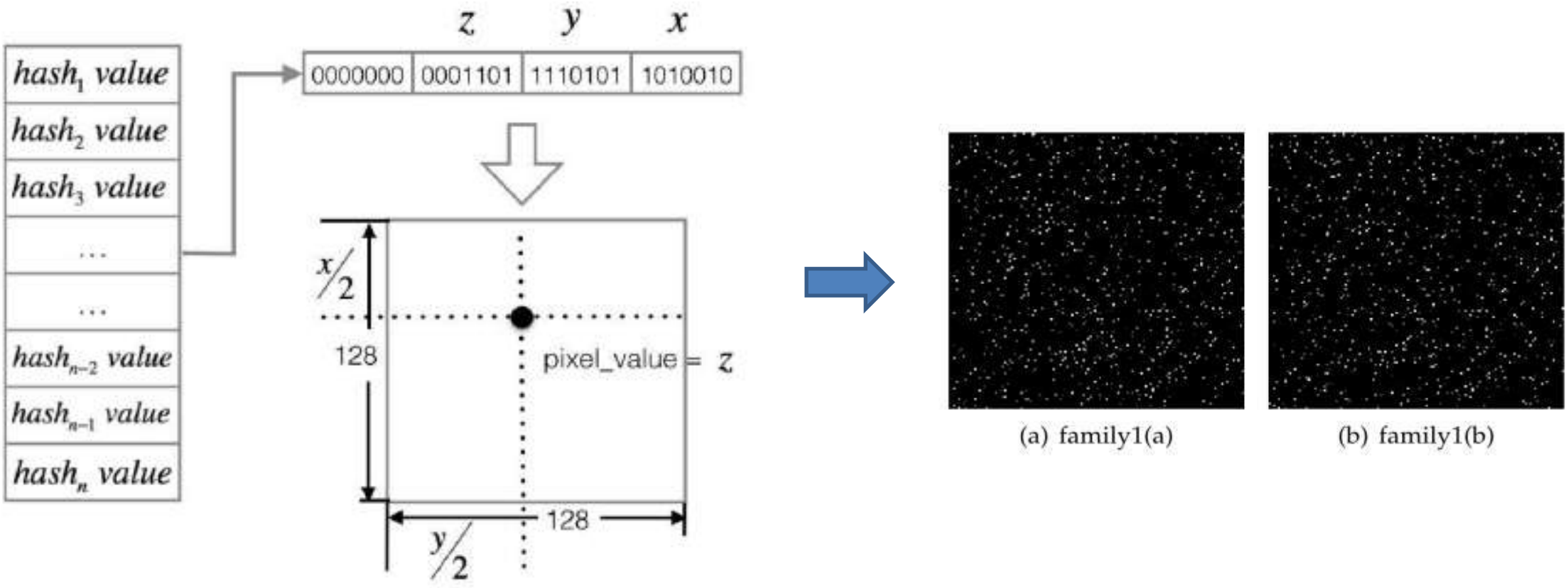
Definition 2.4 (Horizontal visibility graph). Given a time series $T = (v_1, \ldots, v_n)$, its HVG representation $\bar{G} = (V, E)$ has n vertices: $V = (1, \ldots, n)$. An edge $e = (i, j) \in E$ iff $\forall k$ such that $i < k < j$ $(1 \leq i, j \leq n)$ inequality $v_i, v_j > v_k$ is satisfied.





算法原理

- 序列数据转换为Image
  - 时间序列转换为灰度图
  - **Deep Learning and Visualization for Identifying Malware Families**



应用总结

应用总结

- 系统调用序列
  - 入侵攻击检测
  - 恶意程序检测
  - 攻击类型分析
- 系统运行日志
  - 系统运行异常检测
  - 高级持续性威胁检测
  - 攻击定位与溯源

应用总结

- 拓展方向
  - 虚拟化云平台的**VMM**层安全
    - 针对**Hypervisor**的漏洞检测
    - 恶意程序检测
  - 虚拟化云平台内、外部网络安全
    - 网络流量数据
    - 网络功能虚拟化（**NFV**）、软件定义网络（**SDN**）等的安全问题
  - 边缘云计算、雾计算等
    - 物联网网络安全
    - 工业互联网、工控网络安全
  - 不同结构、内容数据之间的转换
    - 序列到**Graph/Image**

……

参考文献

[1] Mishra P, Pilli E S, Varadharajan V, et al. Intrusion detection techniques in cloud environment: A survey[J]. Journal of Network and Computer Applications, 2017, 77: 18-47.

[2] Mishra P, Varadharajan V, Pilli E S, et al. Vmguard: A vmi-based security architecture for intrusion detection in cloud environment[J]. IEEE Transactions on Cloud Computing, 2018, 8(3): 957-971.

[3] Mora-Gimeno F J, Mora-Mora H, Volckaert B, et al. Intrusion Detection System Based on Integrated System Calls Graph and Neural Networks[J]. IEEE Access, 2021, 9: 9822-9833.

[4] Grimmer M, Röhling M M, Kricke M, et al. Intrusion detection on system call graphs[J]. Sicherheit in vernetzten Systemen, pages G1–G18, 2018.

[5] Sun G, Qian Q. Deep learning and visualization for identifying malware families[J]. IEEE Transactions on Dependable and Secure Computing, 2018, 18(1): 283-295.

参考文献

[6] Yin X, Chen X, Chen L, et al. Extension of research on security as a service for VMs in IaaS platform[J]. Security and Communication Networks, 2020, 2020.

[7] Li D, Lin J, Bissyande T F D A, et al. Extracting statistical graph features for accurate and efficient time series classification[C]. 21st International Conference on Extending Database Technology. 2018.

[8] Liu M, Xue Z, Xu X, et al. Host-based intrusion detection system with system calls: Review and future trends[J]. ACM Computing Surveys (CSUR), 2018, 51(5): 1-36.

[9] Qian Q, Li J, Cai J, et al. An anomaly intrusion detection method based on PageRank algorithm[C]. 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing. IEEE, 2013: 2226-2230.

[10] Liu J, Li J, Wu C. An Efficient Massive Log Discriminative Algorithm for Anomaly Detection in Cloud[C]. 2019 IEEE Global Communications Conference (GLOBECOM). IEEE, 2019: 1-6.

大成若缺，其用不弊。
大盈若冲，其用不穷。
大直若屈。大巧若拙。
大辩若讷。静胜躁，寒胜热。清静为天下正。

谢谢！