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

操作系统内核漏洞环境自动生成方法

硕士研究生 杨语航

2026年09月06日

问题回溯

- 相关内容
  - 2026.09.01 王怡男《大模型驱动的软件漏洞PoC自动生成方法》
  - 2026.03.16 杨语航《基于智能体的自动化漏洞重现方法》

内容提要

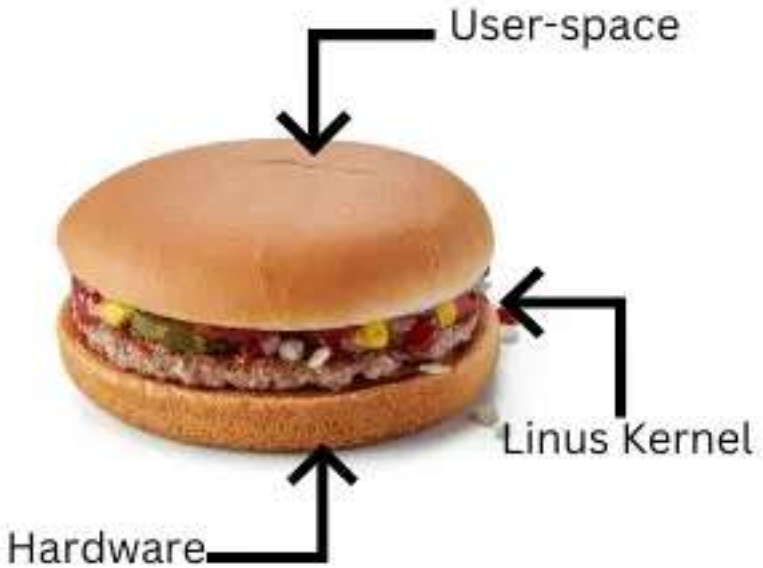
- 预期收获
- 题目内涵解析
- 研究背景与意义
- 研究历史与现状
- 知识基础
- 算法原理
  - KernJC
- 特点总结与工作展望
- 参考文献

预期收获

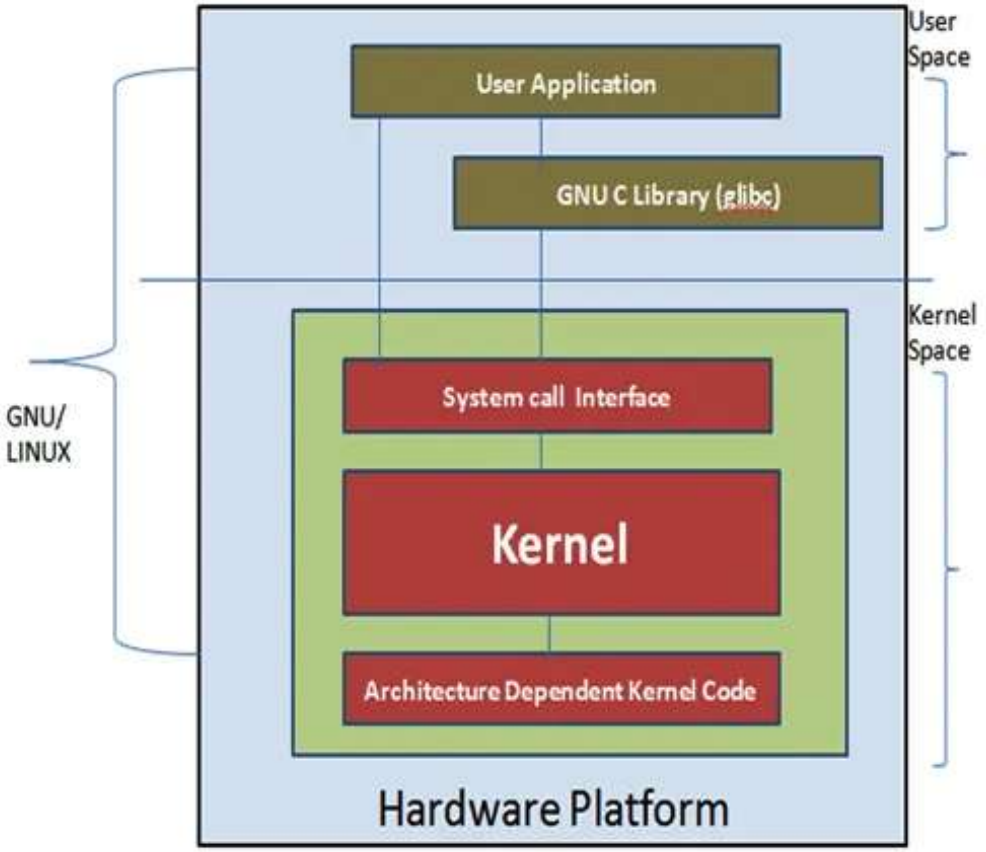
- 预期收获
  - 了解Linux内核漏洞复现的特殊挑战
  - 掌握内核漏洞环境复现中的常用方法和基础概念
  - 理解一种前沿Linux内核漏洞复现技术的核心思想

内涵解析与研究目标

- 题目内涵解析（操作系统内核漏洞环境自动生成方法）
  - 操作系统内核：操作系统中直接管理硬件资源、进程调度、文件系统的核心组件
  - 内核漏洞环境：使目标漏洞真实存在且可触发的运行环境
- 研究目标
  - 面向Linux内核CVE，在给定PoC前提下自动生成能触发目标漏洞的运行环境



内核架构简化版



研究背景与意义 Linux内核漏洞环境复现

- 研究背景
  - 内核安全影响广泛：Linux 内核广泛应用于服务器、移动设备和嵌入式系统，其漏洞可能导致权限提升、信息泄露等严重后果
  - 漏洞环境复现是基础：漏洞影响评估、补丁验证和攻击行为分析，都依赖一个能够真实触发漏洞的运行环境
- 研究意义
  - 提高漏洞验证效率
    - 取代人工复现，减少时间成本
  - 复现标准化
    - 将依赖经验的人工复现转化为可审计扩展的自动化流程

人工复现vs自动化复现工具 belike

研究历史与现状　内核漏洞复现

SyzScope针对内核模糊测试产生的crash缺少安全评估的问题，基于已有PoC和崩溃报告自动分析漏洞影响，挖掘控制流劫持、任意内存写等风险行为，推动从发现崩溃到评估危害转变

SyzBridge针对上游漏洞的PoC在Ubuntu等下游发行版中难以复现的问题，通过分析不同内核版本和发行版的差异，自动适配PoC和复现条件，从而支持跨发行版内核的漏洞可利用性评估

Patch-to-PoC关注内核N-day漏洞从补丁到PoC的自动化复现问题，构建一种具备代码浏览、虚拟机管理、交互执行和调试能力的Agent系统，以安全补丁为输入自动生成能够触发漏洞的 PoC

2022　2024　2026

2023

SyzDirect针对内核漏洞复现和补丁测试中目标路径难以到达的问题，将DGF应用于操作系统内核，通过静态分析识别到达补丁位置所需的系统调用和参数约束，引导模糊测试更高效地触发目标漏洞代码

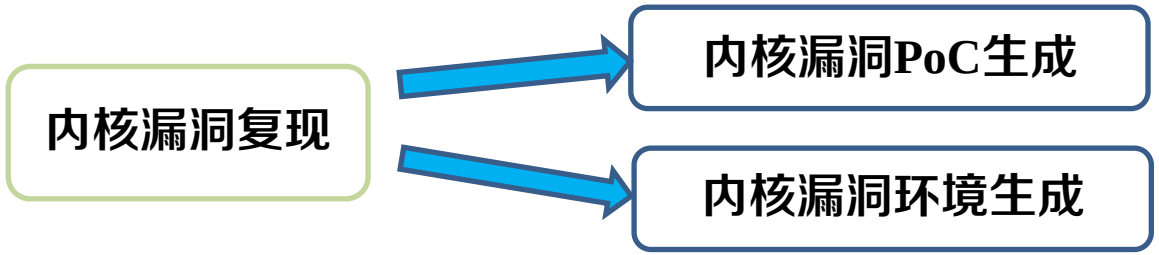
2024

KernJC关注内核漏洞复现中运行环境构建被忽视的问题，借助补丁分析识别真实包含漏洞的内核版本，并通过图分析推断触发漏洞所需的关键编译配置，自动生成可复现内核漏洞环境

2026

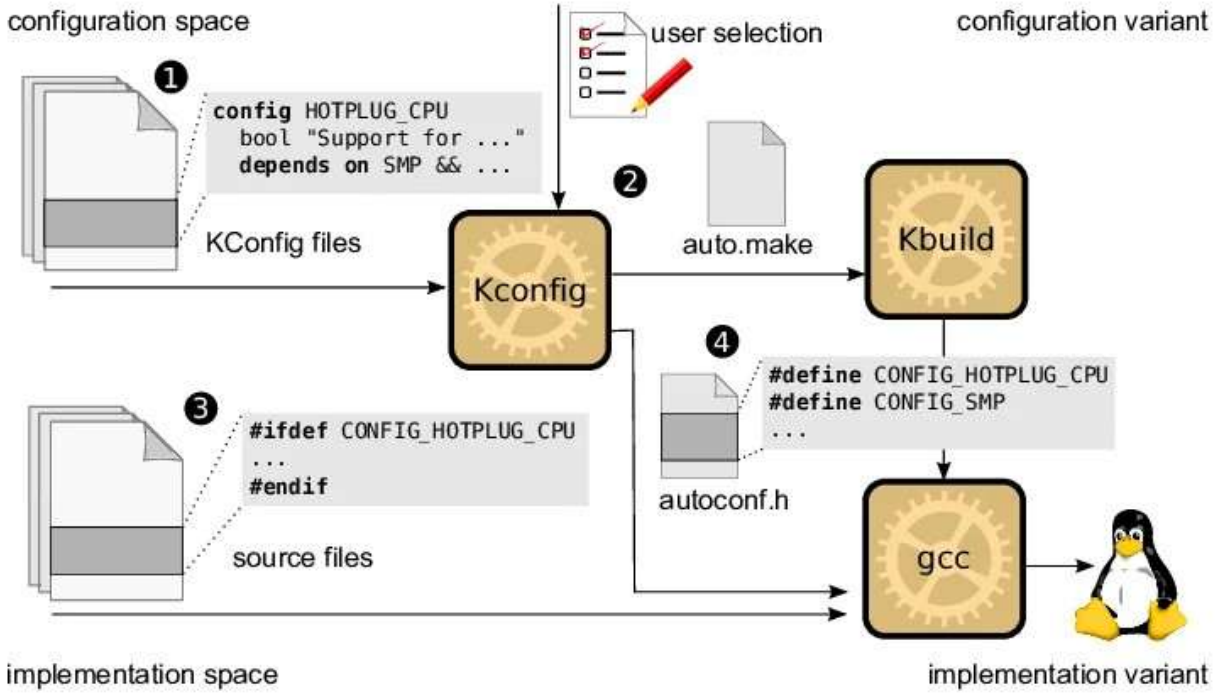
FCC关注真实生产内核中指定CVE是否可触发的问题，从漏洞描述、补丁和代码中提取相关配置项，并根据Linux 内核配置规则自动补全依赖关系，再通过编译运行和触发验证去删除冗余配置，得到触发漏洞的最小配置集

近年从内核漏洞挖掘的研究主线中衍生出一条支线——内核漏洞自动复现



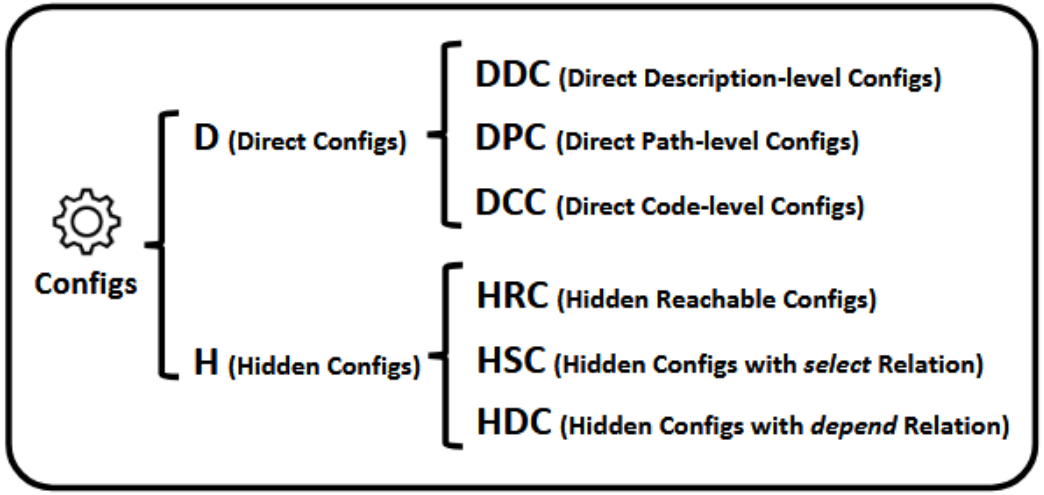
知识基础

- 内核版本
  - Linux 内核版本长期演进且分支众多
- 编译配置文件
  - Kconfig files：可以选的配置
  - User selection：用户选择的配置
  - Source files：基于已选配置编译代码块
  - gcc：编译



该漏洞影响的内核版本v5.18~v7.1-rc6

- 内核配置依赖关系（给定待分析的漏洞）
  - 直接配置
    - 直接从漏洞描述、补丁、源码中分析识别的配置
  - 隐藏配置
    - 不直接出现在漏洞代码附近，但会影响直接配置是否可以生效的配置
- 两种简单的内核编译配置形式
  - defconfig
    - Linux内核提供的默认配置
  - allyesconfig
    - 开启全部配置



【RAID】

KernJC: Automated Vulnerable Environment Generation for Linux Kernel Vulnerabilities

KernJC TIPO

T	目标	自动生成能够复现指定Linux CVE的漏洞环境
I	输入	CVE信息*n、补丁*1、PoC*1、内核源码*1
P	处理	1. 漏洞信息收集：CVE描述、补丁、PoC 2. 漏洞版本识别：通过补丁匹配回溯真实易受影响的内核版本 3. 漏洞配置识别：结合源码路径、条件编译和Kconfig图推断必要配置 4. 漏洞环境生成：叠加必要配置，编译内核，部署虚拟机环境
O	输出	目标CVE真实影响版本*n、内核配置文件*1、可运行的漏洞环境*1
P	问题	1.漏洞数据库中公开的影响版本范围不准确 2.内核配置项数量庞大，依赖复杂，人工分析必要配置成本高
C	条件	公开CVE信息、漏洞相关的修复补丁、可用于验证的PoC
D	难点	1. 如何准确识别真实存在漏洞的内核版本 2. 如何保证推断触发漏洞所需的内核配置的完整性
L	水平	RAID 2024（CCF-A）

- 漏洞版本真实性
  - 科学问题：漏洞存在性需要在版本演化序列中，通过补丁前后代码状态变化进行判定
  - 应用问题：漏洞数据库中的受影响版本范围可能不准确，导致复现时选中了已修复或并不含漏洞的内核版本
- 漏洞配置可达性
  - 科学问题：漏洞代码的可达性不是单一配置项决定的，而是由源码、编译条件和配置约束等共同形成的高维依赖关系决定
  - 应用问题：默认配置可能无法启用漏洞相关代码，而全部开启配置又会引入冲突、不可启动等情况

- 如何验证漏洞代码是否真实存在于当前版本呢?
  - 思路：将版本是否受影响转化为补丁是否已经存在的代码状态判定问题

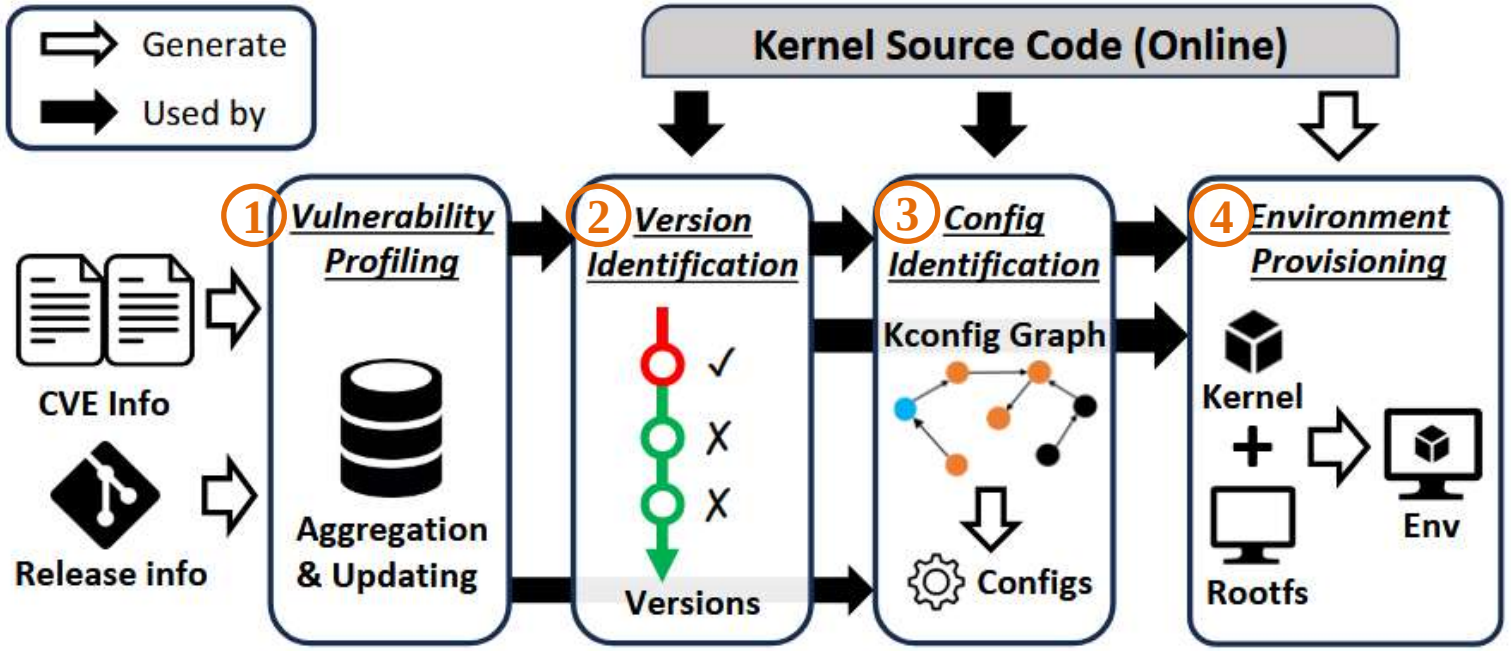
mainline:	7.3-rc1	2026-08-30	[tarball]		[patch]		[view diff]	[browse]
stable:	7.2.3	2026-09-02	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse] [changelog]
stable:	7.1.13 [EOL]	2026-09-02	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse] [changelog]
longterm:	6.18.49	2026-09-02	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse] [changelog]
longterm:	6.12.108	2026-09-02	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse] [changelog]
longterm:	6.6.156	2026-09-02	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse] [changelog]
longterm:	6.1.187	2026-09-02	[tarball]	[pgp]	[patch]	[inc. patch]	[view diff]	[browse] [changelog]

- 如何找到触发目标漏洞所需的全部内核配置呢?
  - 观察：相关功能是否被编译进内核，不仅取决于某个特定配置项，还取决于一系列存在依赖和选择关系的配置项
  - 思路：将配置之间的依赖和选择关系抽象为可分析的结构，从直接配置出发推断隐藏配置，最终得到激活漏洞代码所需的充分配置集合

关于漏洞配置推断，首先要考虑的是充分性，其次才是必要性

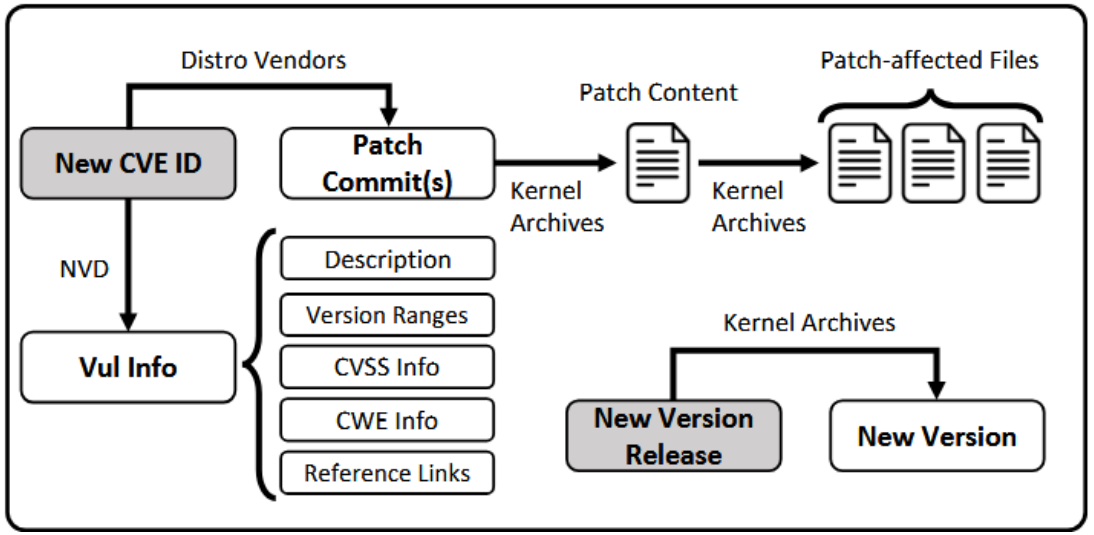
KernJC 算法原理图

- 思路分析
  - ① 收集CVE、补丁和内核版本信息
  - ② 基于补丁存在性检测，回溯识别真实易受影响版本
  - ③ 分析源码路径、条件编译和配置依赖，推断必要配置
  - ④ 编译目标内核并部署虚拟机，生成可验证漏洞环境

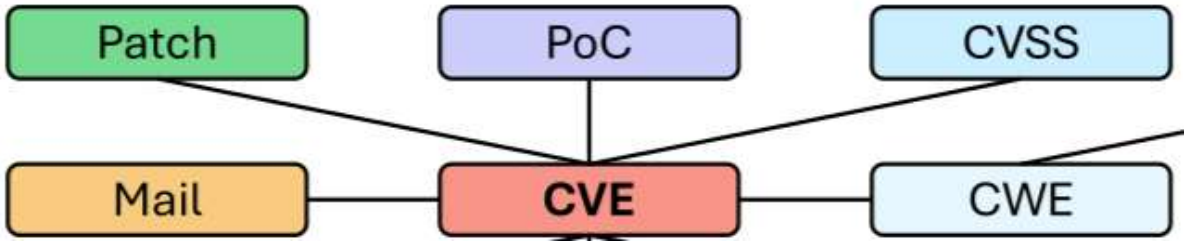


算法原理 Vulnerability Profiling

- **Vulnerability Profiling**
  - 检索方式：给定CVE编号，从公开漏洞数据库和发行版公告中检索信息
  - 检索内容
    - 漏洞信息：漏洞类型、描述、影响版本等
    - 补丁信息：补丁内容、修改文件、相关函数
    - 内核版本发行信息
    - PoC：用于后续验证复现环境

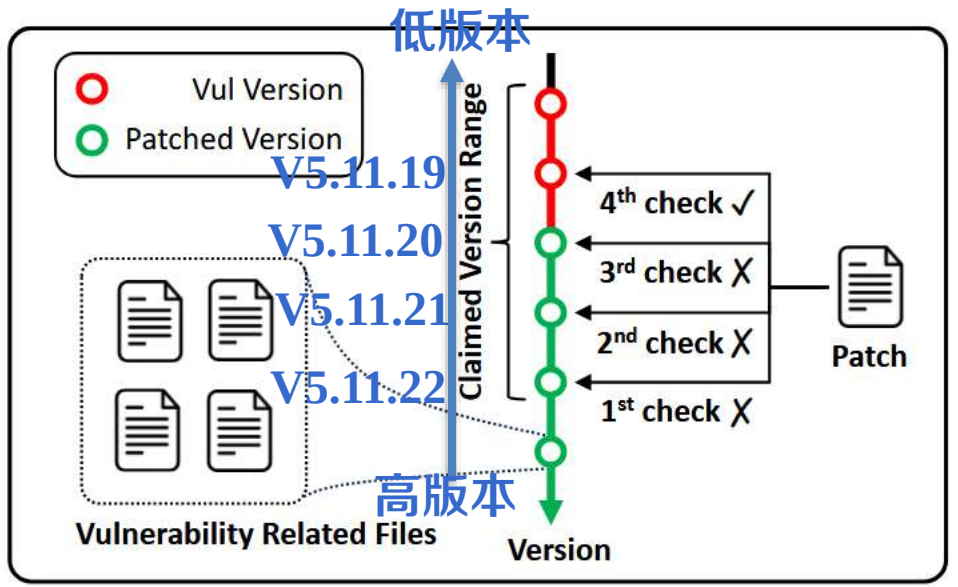


为后续补丁分析和配置识别收集原始信息



算法原理 Version Identification

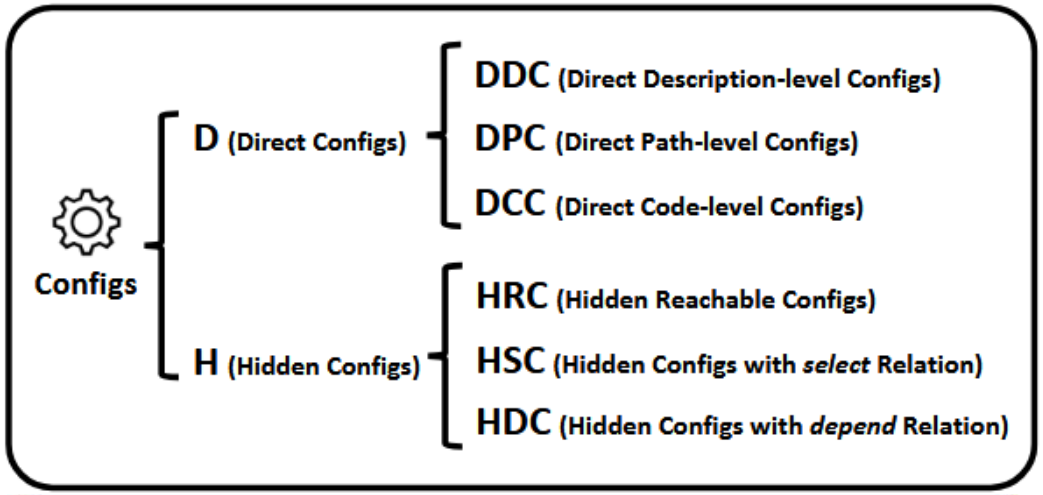
- **Version Identification**
  - 核心思想：将版本是否受影响转化为修复补丁是否存在的问题
  - 处理过程
    - 将CVE声明的影响版本范围映射到最新的发布序列
    - 从影响的最高版本向低版本，对目标版本应用修复补丁，然后检查补丁状态
    - 补丁已存在
      - 该版本已经修复，声明可能误报，继续回溯
    - 补丁不存在
      - 该版本可能包含漏洞，可作为复现候选版本



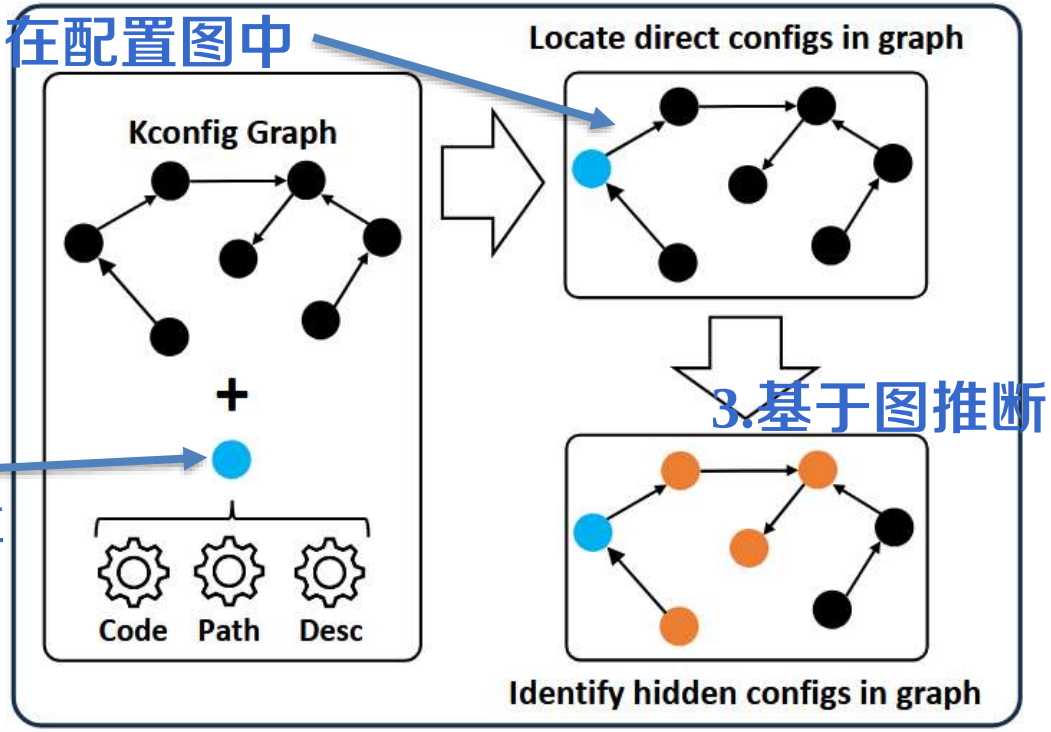
假设初次应用补丁就成功了，是否需要向更高版本追溯呢?

算法原理 Config Identification

- 充分配置推断
  - 直接配置
    - 描述级配置DDC
    - 路径级配置DPC
    - 代码级配置DCC
  - 隐藏配置
    - 可达配置HRC
    - 依赖关系配置HDC
    - 选择关系配置HSC



2.将直接配置放在配置图中

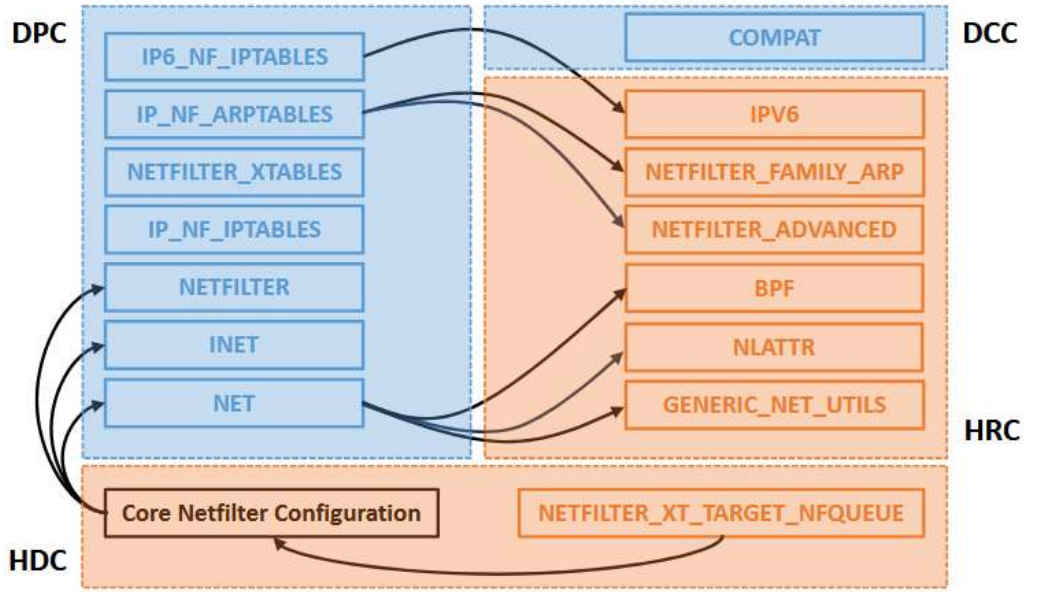


3.基于图推断隐藏配置

1.先推断出直接配置

算法原理 Config Identification

- **Kconfig Graph构建**
  - 从内核源码根Kconfig文件开始，递归解析子目录Kconfig
  - 节点
    - 内核配置文件中的配置项
    - 例如：CONFIG NETFILTER_XTABLES
  - 有向边
    - 配置项之间的依赖关系、选择关系等配置关系
    - 依赖关系，例如： CONFIG NETFILTER_XTABLES depends on NET && ...
    - 选择关系，例如： CONFIG A select B

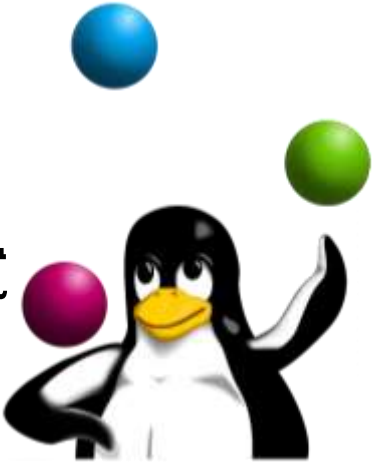


算法原理 Environment Provisioning

- 环境构建
  - 拉取内核：具体的内核版本由版本识别阶段确定
  - 生成配置文件：合并defconfig和针对漏洞触发推断得到的充分配置

```
root@android:~/kernel/linux-4.14.15# cat .config |grep KVM
CONFIG_KVM_GUEST=y
CONFIG_KVM_DEBUG_FS=y
CONFIG_PTP_1588_CLOCK_KVM=m
CONFIG_DRM_I915_GVT_KVMGT=m
CONFIG_HAVE_KVM=y
# CONFIG_KVM_MMU_AUDIT is not set
```

  - 内核编译：基于生成配置文件进行编译，得到包含目标漏洞的内核镜像
  - 构建虚拟机：结合内核镜像部署QEMU/KVM虚拟机，用于后续运行测试
- 环境验证
  - 执行PoC，分析运行和崩溃日志，判断漏洞环境是否构建成功



实验设计　数据资源

- 实验配置
  - 原始数据集：从NVD收集Linux内核CVE
  - 复现验证基：去除无效CVE和缺少补丁提交的漏洞后，筛选66个漏洞和PoC集合
  - 硬件配置：Intel®Xeon ® E5-2640 V4 @ 2.40GHz CPU和128G内存
  - 宿主机操作系统：Ubuntu 22.04.3 LTS 64 bit OS
  - 兼容性处理：部分漏洞在 Ubuntu 18.04 容器中实验，避免编译器版本与旧内核源码不兼容
- 对比方法
  - KernJC：推断的漏洞版本 + defconfig配置 + 分析的额外配置项
  - 默认配置基线：公开声明漏洞版本 + defconfig配置
- 验证方法
  - 在两类环境中分别运行PoC验证，分析运行和崩溃报告

- RQ1：KernJC的漏洞复现能力如何?
- 实验对象
  - 66个真实Linux内核漏洞，均配有可运行PoC
- 实验结果
  - KernJC复现成功率100%
  - 4个漏洞存在NVD错误版本声明
  - 32个漏洞需要非默认配置
  - 34个漏洞无法直接依赖声明版本 + 默认配置复现
- 结论
  - KernJC能有效生成真实漏洞复现环境，并降低版本选择和配置选择带来的复现失败风险

ID	Type	CVSS	Subsystem	RwKC?	RwDC?	FPV?
CVE-2016-10150	UAF	9.8	virt/kvm	✓	✗	✗
CVE-2016-4557	UAF	7.8	kernel/bpf	✓	✗	✗
CVE-2016-6187	OOB	7.8	security/apparmor	✓	✗	✗
CVE-2017-16995	Logic	7.8	kernel/bpf	✓	✗	✗
CVE-2017-18344	OOB	5.5	kernel/time	✓	✗	✗
CVE-2017-2636	DF	7.0	drivers/tty	✓	✗	✗
CVE-2017-6074	DF	7.8	net/dccp	✓	✗	✗
CVE-2017-8824	UAF	7.8	net/dccp	✓	✗	✗
CVE-2018-12233	OOB	7.8	fs/jfs	✓	✗	✗
CVE-2018-5333	ND	5.5	net/rds	✓	✗	✗
CVE-2018-6555	UAF	7.8	net/irda	✓	✗	✗
CVE-2019-6974	UAF	8.1	virt/kvm	✓	✗	✗
CVE-2020-14381	UAF	7.8	futex	✓	✓	✓
CVE-2020-16119	UAF	7.8	net/dccp	✓	✗	✗
CVE-2020-25656	UAF	4.1	drivers/tty	✓	✓	✓
CVE-2020-25669	UAF	7.8	drivers/input	✓	✗	✗
CVE-2020-27194	OOB	5.5	kernel/bpf	✓	✗	✗
CVE-2020-27830	ND	5.5	drivers/accessibility	✓	✗	✗
CVE-2020-28941	ND	5.5	drivers/accessibility	✓	✗	✗
CVE-2020-8835	OOB	7.8	kernel/bpf	✓	✗	✗
CVE-2021-22555	OOB	7.8	net/netfilter	✓	✗	✓
CVE-2021-26708	UAF	7.0	net/vmw_vsock	✓	✗	✗
CVE-2021-27365	OOB	7.8	drivers/scsi	✓	✗	✗
CVE-2021-34866	OOB	7.8	kernel/bpf	✓	✗	✗
CVE-2021-3490	OOB	7.8	kernel/bpf	✓	✗	✗
CVE-2021-3573	UAF	6.4	net/bluetooth	✓	✗	✓
CVE-2021-42008	OOB	7.8	drivers/net	✓	✗	✗
CVE-2021-43267	OOB	9.8	net/tipc	✓	✗	✗
CVE-2022-0995	OOB	7.8	watch_queue	✓	✗	✗
CVE-2022-1015	OOB	6.6	net/netfilter	✓	✗	✗
CVE-2022-25636	OOB	7.8	net/netfilter	✓	✗	✗
CVE-2022-32250	UAF	7.8	net/netfilter	✓	✗	✗
CVE-2022-34918	OOB	7.8	net/netfilter	✓	✗	✗
CVE-2023-32233	UAF	7.8	net/netfilter	✓	✗	✗

列出34个非默认情况

- RQ2：KernJC识别的内核配置是否真的有助于漏洞复现?
- 实验对象
  - 32个依赖非默认配置的漏洞
- 实验结果
  - 16个漏洞的复现依赖隐藏配置
  - 主要集中在eBPF和Netfilter等子系统
- 结论
  - 隐藏配置识别是许多复杂内核子系统漏洞成功复现的关键

CVE	Subsystem	Kernel	Kconfig Graph	DDC	DPC	DCC	HRC	HSC	HDC
CVE-2016-10150	virt/kvm	v4.8.12	12337v+38250e	0	1	0	39	0	4
CVE-2016-4557	kernel/bpf	v4.5.4	11845v+36556e	0	1	0	0	2	0
CVE-2016-6187	security/apparmor	v4.6.4	12021v+37152e	0	1	0	14	0	2
CVE-2017-16995	kernel/bpf	v4.14.8	13436v+41928e	0	1	0	0	2	0
CVE-2019-6974	virt/kvm	v4.20.7	14054v+44510e	0	1	0	42	0	4
CVE-2020-27194	kernel/bpf	v5.8.14	15340v+50234e	0	1	0	0	2	1
CVE-2020-8835	kernel/bpf	v5.6	14957v+48344e	0	1	0	0	2	1
CVE-2021-22555	net/netfilter	v5.11.14	15808v+51956e	0	7	1	10	3	406
CVE-2021-34866	kernel/bpf	v5.13.13	15982v+52565e	0	1	0	0	2	3
CVE-2021-3490	kernel/bpf	v5.12.3	15855v+52142e	0	1	0	0	2	2
CVE-2021-3573	net/bluetooth	v5.12.9	15851v+52148e	0	1	0	32	0	45
CVE-2022-1015	net/netfilter	v5.16.17	16244v+53665e	0	1	0	4	0	241
CVE-2022-25636	net/netfilter	v5.16.11	16243v+53663e	0	4	0	19	2	241
CVE-2022-32250	net/netfilter	v5.18.1	16542v+54754e	0	1	0	4	0	238
CVE-2022-34918	net/netfilter	v5.18.10	16548v+54770e	0	1	0	4	0	238
CVE-2023-32233	net/netfilter	v6.3.1	17120v+57166e	0	2	0	5	0	317

但是KernJC识别的隐藏配置中其实存在很多冗余

- RQ3：KernJC能否发现NVD中错误的受影响版本声明?
- 实验对象
  - 2256个带补丁提交的Linux内核CVE
- 实验结果
  - 128个CVE存在错误版本声明
  - 总共识别3042个错误标记版本
- 结论
  - 内核版本识别是自动化漏洞环境生成中的必要步骤

CVE	CVSS	FP Version Range	Vulnerable Version	FP Count
CVE-2017-1000407	7.4	v4.14.6 – v4.14.325	v4.14.5	320
CVE-2017-18216	5.5	v4.14.57 – v4.14.325	v4.14.56	269
CVE-2017-18224	4.7	v4.14.57 – v4.14.325	v4.14.56	269
CVE-2020-35508	4.5	v5.9.7 – v5.11.22	v5.9.6	229
CVE-2021-4002	4.4	v5.15.5 – v5.15.132	v5.15.4	128
CVE-2021-4090	7.1	v5.15.5 – v5.15.132	v5.15.4	128
CVE-2022-0264	5.5	v5.15.11 – v5.15.132	v5.15.10	122
CVE-2021-4155	5.5	v5.15.14 – v5.15.132	v5.15.13	119
CVE-2016-10906	7.0	v4.4.191 – v4.4.302	v4.4.190	112
CVE-2015-4170	4.7	v3.12.7 – v3.13.3	v3.12.6	72

算法总结 KernJC

- 算法流程
  - 收集漏洞信息，构建漏洞画像
  - 基于补丁存在性检测，定位真实易受影响版本
  - 通过Kconfig图，推断必要配置
  - 编译目标内核并部署虚拟机，生成可运行复现环境
- 算法优势
  - 工具流程完整，支持从CVE到可验证环境的自动化
  - 能自动补充非默认配置，对复杂内核子系统漏洞复现有一定优势
- 算法不足
  - 依赖较多原始信息
  - 配置推断可能引入冗余配置

特点总结与未来展望

特点总结与未来展望

- 特点总结
  - 从PoC生成转向漏洞环境构建
  - 面向公开CVE、真实内核版本和已有PoC，验证了漏洞环境构建的价值
- 未来展望
  - 基于近三年内核CVE，更新更大规模的复现数据集
  - 编译配置从充分性提升到必要性

参考文献

[1] Ruan B, Liu J, Zhang C, et al. Kernjc: Automated vulnerable environment generation for linux kernel vulnerabilities[C]. Proceedings of the 27th International Symposium on Research in Attacks, Intrusions and Defenses. 2024: 384-402.

知人者智，自知者明。胜人者有力，自胜者强。知足者富。强行者有志。不失其所者久。死而不亡者，寿。

谢谢！