KernJC: Automated Vulnerable Environment Generation for Linux Kernel Vulnerabilities

Bonan Ruan, Jiahao Liu, Chuqi Zhang, Zhenkai Liang

RAID, September 2024
Padua, Italy

NUS
National University of Singapore

Impact of Linux Kernel Vulnerabilities

- Privilege escalation on servers
- Android rooting
- Container escaping

Source: generated by ChatGPT

Active Exploitation Observed for Linux Kernel Privilege Escalation Vulnerability (CVE-2024-1086)

June 6, 2024 | Adam Cardillo - Amit Serper - Suraj Sahu | Cloud and Application Security • Exposure Management

Home > News > Security > Google fixes Android kernel zero-day exploited in targeted attacks

Google fixes Android kernel zero-day exploited in targeted attacks

By Sergiu Gatlan August 5, 2024 06:40 PM 1

Linux Kernel Bug Allows Kubernetes Container Escape

January 31, 2022 Container Linux, container security, container vulnerability, kubernetes, Linux kernel

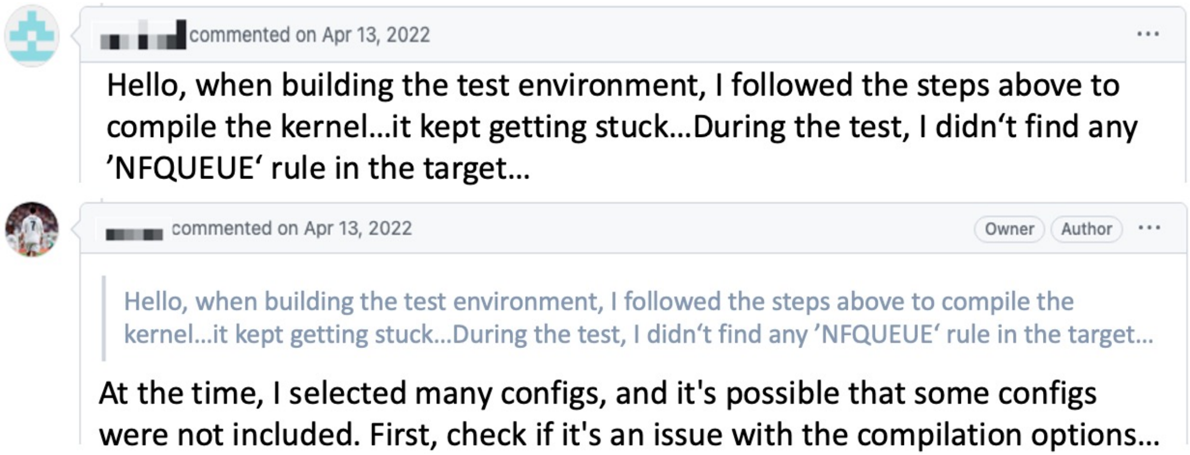
by Nathan Eddy

Kernel Vulnerability Reproduction

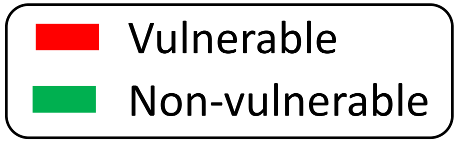
- **Reproduction is pivotal to the comprehension of vulnerabilities.**
- **Application Scenarios:**
  - Vulnerability severity assessment
  - Design of detection and mitigation
  - Evaluation of detection and mitigation
- **Two crucial elements** for reproduction:
  - The vulnerable environment
  - The Proof of Concept (PoC)
- Existing studies focus on PoC generation, **while the generation of reproduction environment is overlooked, but non-trivial.**

Challenges

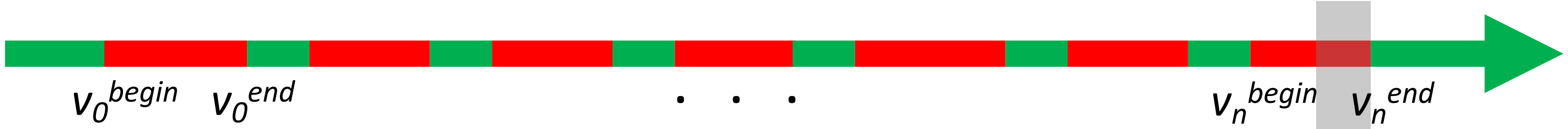
- **Incorrect vulnerable versions:**
  - It is hard to guarantee that the selected kernel version is vulnerable, as the vulnerability version claims in online databases are occasionally incorrect.
- **Intricate kernel configs:**
  - For many kernel vulnerabilities, <u>intricate non-default kernel configs</u> must be set to include and trigger these vulnerabilities, while less information is available on how to recognize these configs.



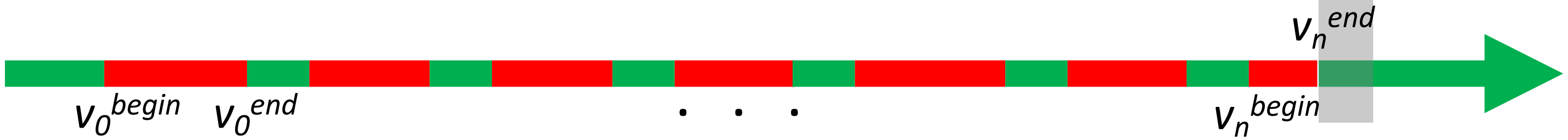
Example: CVE-2021-22555 (OOB in Netfilter)



- Vulnerable version ranges claimed by NVD:



- Actually, some versions have already been patched:



- Kernel configs needed for triggering this vulnerability:

CONFIG_COMPAT	CONFIG_NETFILTER_XTABLES	CONFIG_NETFILTER
CONFIG_NET	CONFIG_NETFILTER_FAMILY_ARP	CONFIG_NETFILTER_ADVANCED
CONFIG_INET	CONFIG_IP_NF_IPTABLES	CONFIG_NLATTR
CONFIG_IPV6	CONFIG_IP_NF_ARPTABLES	CONFIG_GENERIC_NET_UTILS
CONFIG_BPF	CONFIG_IP6_NF_IPTABLES	CONFIG_NETFILTER_XT_TARGET_NFQUEUE

Given a kernel vulnerability, how can we identify the real vulnerable version and necessary configs?

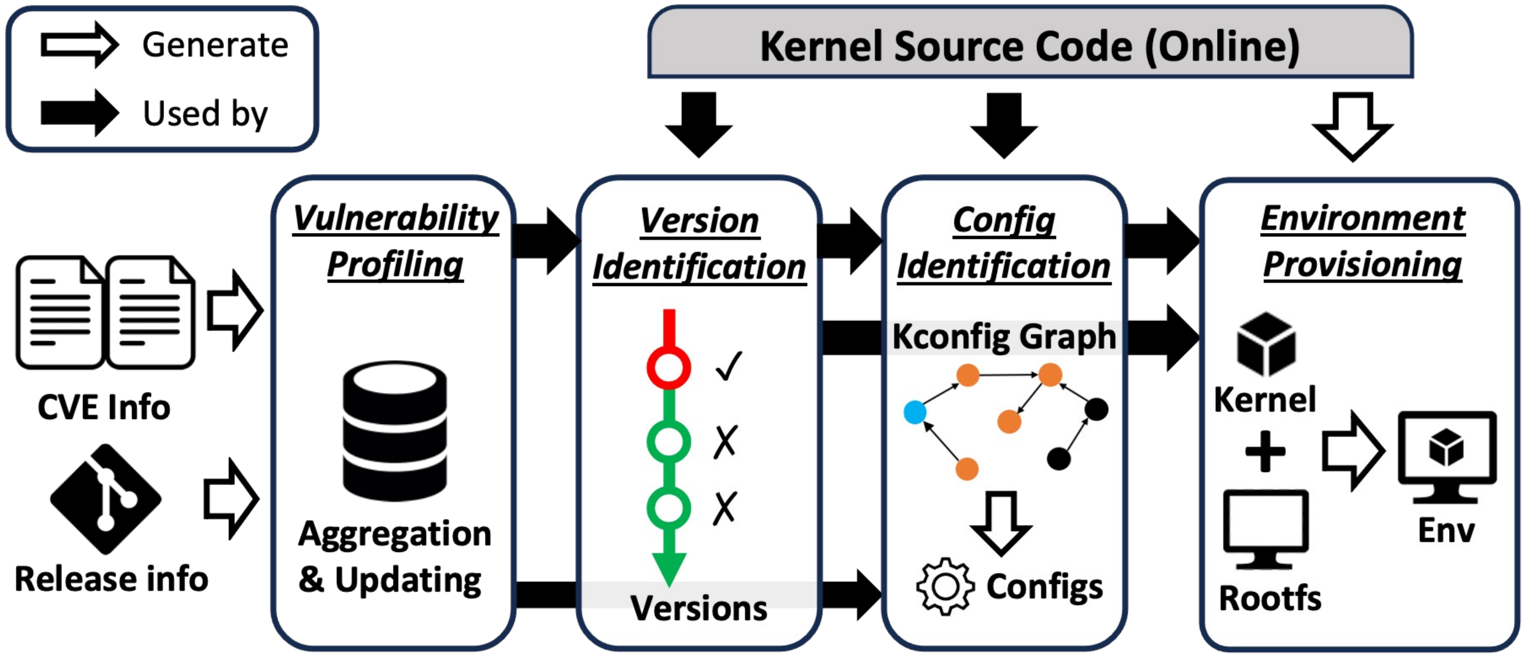
Observations

- The presence of patch implies the absence of vulnerability.
- Kconfig and Kbuild mechanisms work in tandem to tailor the kernel.
- Kernel configs can be regarded as graph.

Insights

- Given a kernel version, check the presence of patch.
- Parse the Kconfig and Makefile files into a graph.
- Abstract the config identification problem into a graph searching problem.

Overview of KernJC



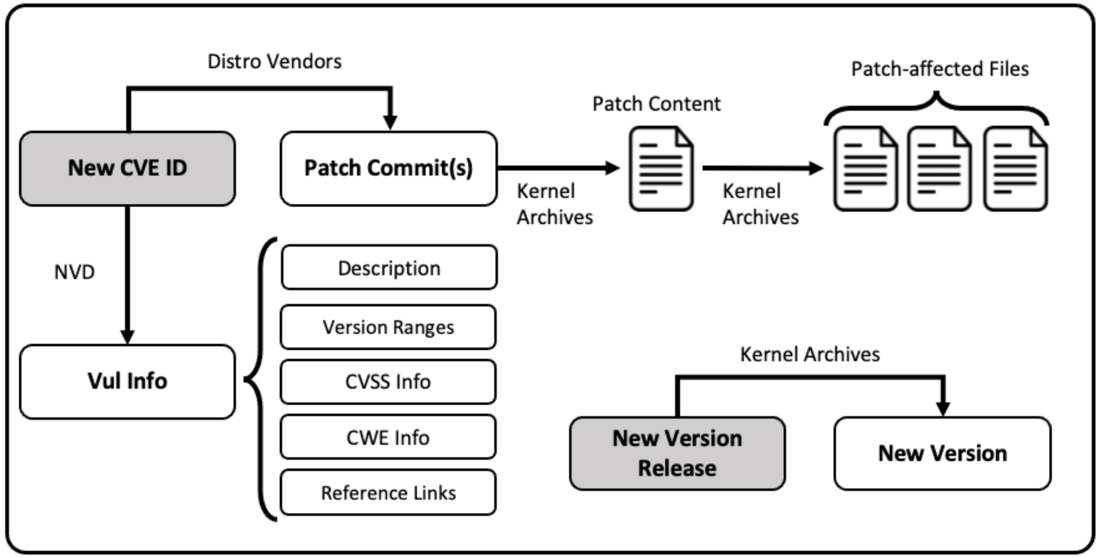
- **Vulnerability Profiling:** Collect vulnerability information for later usage.
- **Version Identification:** Perform patch operation to detect patch presence.
- **Config Identification:** Build Kconfig graph and mine reachable configs.
- **Environment Provisioning:** Build the kernel and provision the virtual machine.

Vulnerability Profiling

Config Identification

- CVE descriptions
- Vulnerable version ranges
- Patch commit(s) and contents
- Files affected by patches
- Linux kernel version release list

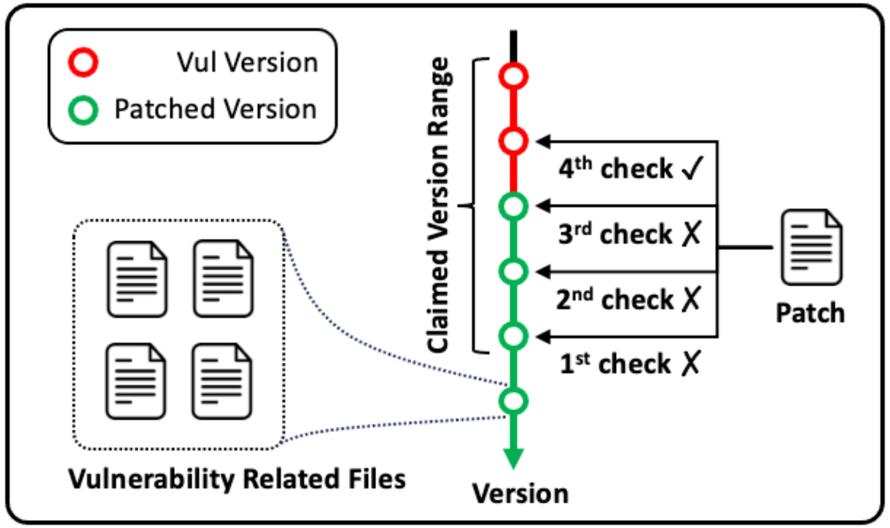
Version Identification



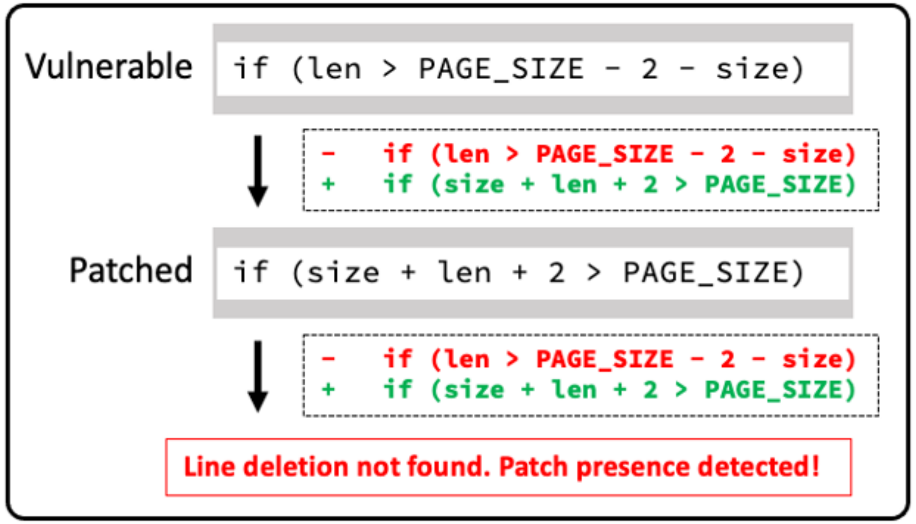
Incremental Aggregation & Updating

Vulnerability Version Identification

- Locate the latest vulnerable version *v* claimed by NVD.
- Start from *v* and move downwards along the kernel version list:
  - Apply the patch on vulnerability related files of each version.
  - Stop when no patch presence detected.



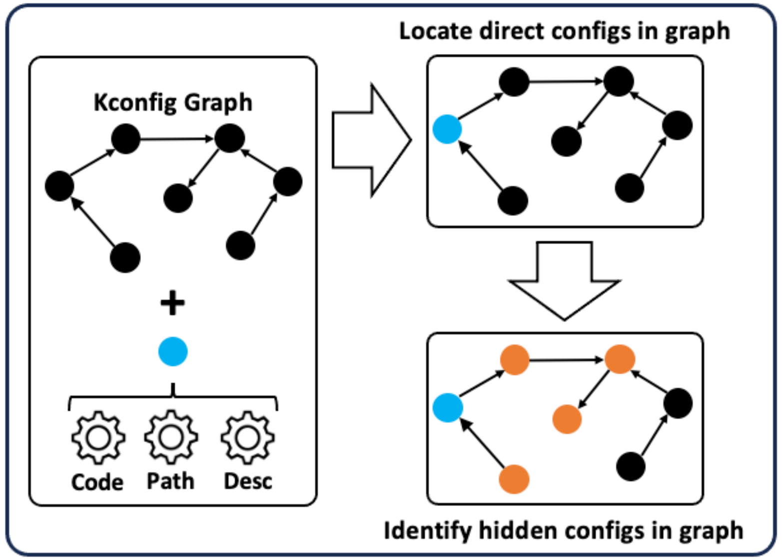
Identification Process

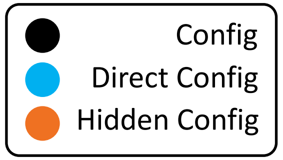


Identification Example

Vulnerability Config Identification

- Build the Kconfig graph for target kernel.
- Gather *direct configs* (*D = DDC* ∪ *DPC* ∪ *DCC*):
  - DDC: Direct Description-level Configs
  - DPC: Direct Path-level Configs
  - DCC: Direct Code-level Configs
- For each config *c* in *D*:
  - Locate *c* in the Kconfig graph.
  - Discover hidden configs for *c* (H_c = *HRC* ∪ *HSC* ∪ *HDC*):
    - HRC: Hidden Reachable Configs from *c*
    - HSC: Hidden Configs with *Select* relation to *c*
    - HDC: Hidden Configs with *Depend* relation *c*
- Collect all *hidden configs*.
- Final result = *D* ∪ *H*.





Evaluation

- **Research Questions:**
  - RQ1: How is KernJC’s **performance in reproduction** of kernel vulnerabilities?
  - RQ2: How well do the **configs identified by KernJC** facilitate the reproduction of kernel vulnerabilities?
  - RQ3: How many **incorrect version claims in NVD** can KernJC detect for Linux kernel vulnerabilities?
- **Dataset:**
  - RQ1 & RQ2: 66 real-world kernel CVEs with workable PoCs
    - CVEs are collected from relevant research published on top security conferences in the past five years.
    - PoCs are collected from the Internet and modified to make them workable.
  - RQ3: 2,256 kernel CVEs with associated patches

Performance in Reproduction

- KernJC successfully builds effective reproduction environments for all 66 vulnerabilities.
  - 4 of 66 are detected to have incorrect (FP) version claims in NVD.
  - 32 of 66 need non-default configs identified by KernJC to be activated.

CVE	RwKC?	RwDC?	FPV?	CVE	RwKC?	RwDC?	FPV?	CVE	RwKC?	RwDC?	FPV?	CVE	RwKC?	RwDC?	FPV?
2016-10150	✓	✗	✗	2018-12233	✓	✗	✗	2020-27194	✓	✗	✗	2021-3490	✓	✓	✗
2016-4557	✓	✗	✗	2018-5333	✓	✗	✗	2020-27830	✓	✗	✗	2021-3573	✓	✗	✓
2016-6187	✓	✗	✗	2018-6555	✓	✗	✗	2020-28941	✓	✗	✗	2021-42008	✓	✗	✗
2017-16995	✓	✗	✗	2019-6974	✓	✗	✗	2020-8835	✓	✗	✗	2021-43267	✓	✗	✗
2017-18344	✓	✗	✗	2020-14381	✓	✓	✓	2021-22555	✓	✗	✓	2022-0995	✓	✗	✗
2017-2636	✓	✗	✗	2020-16119	✓	✗	✗	2021-26708	✓	✗	✗	2022-1015	✓	✗	✗
2017-6704	✓	✗	✗	2020-25656	✓	✓	✓	2021-27365	✓	✗	✗	2022-25636	✓	✗	✗
2017-8824	✓	✗	✗	2020-25669	✓	✗	✗	2021-34866	✓	✗	✗	2022-32250	✓	✗	✗
				2022-34918	✓	✗	✗	2023-32233	✓	✗	✗				

RwKC: Reproducibility with KernJC-identified Configs
RwDC: Reproducibility with Default Configs
FPV: False Positive Version claims in NVD

Configs Identified by KernJC

- Half of the 32 vulnerabilities necessitate HSC or HDC for activation.
  - Consequently, HSC and HDC identified by KernJC play an important role in constructing effective reproduction environments for kernel vulnerabilities.

CVE	DDC	DPC	DCC	HRC	HSC	HDC	CVE	DDC	DPC	DCC	HRC	HSC	HDC
CVE-2016-10150	0	1	0	39	0	4	CVE-2021-34866	0	1	0	0	2	3
CVE-2016-4557	0	1	0	0	2	0	CVE-2021-3490	0	1	0	0	2	2
CVE-2016-6187	0	1	0	14	0	2	CVE-2021-3573	0	1	0	32	0	45
CVE-2017-16995	0	1	0	0	2	0	CVE-2022-1015	0	1	0	4	0	241
CVE-2019-6974	0	1	0	42	0	4	CVE-2022-25636	0	4	0	19	2	241
CVE-2020-27194	0	1	0	0	2	1	CVE-2022-32250	0	1	0	4	0	238
CVE-2020-8835	0	1	0	0	2	1	CVE-2022-34918	0	1	0	4	0	238
CVE-2021-22555	0	7	1	10	3	406	CVE-2023-32233	0	2	0	5	0	317

Vulnerabilities relying on HSC or HDC

Incorrect Version Claims in NVD

- We identify 128 vulnerabilities with incorrect version claims in NVD.
- The aggregate count of incorrect (FP) versions is 3,042.
  - averaging 24 incorrect versions per identified vulnerability.

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

Top 10 vulnerabilities sorted by FP version count

Conclusion

- We point out two challenges in the generation of vulnerable environments for Linux kernel vulnerabilities.
- We propose patch-based and graph-based approaches to solve these challenges.
- **KernJC:** automated vulnerable environment generation for Linux kernel vulnerabilities
  - https://github.com/NUS-CURIOSITY/KernJC

Thank you!

Contact me at r-bonan@comp.nus.edu.sg

iTerm2 Shell Edit View Session Scripts Profiles Toolbelt Window Help 16:47

```
(venv) ➜  KernJC git:(main) ✗ ./kjc build CVE-2021-22555
[*] Building environment for CVE-2021-22555
```