RetSpill: Igniting User-Controlled Data to Burn Away Linux Kernel Protections

Kyle Zeng[1], Zhenpeng Lin[2], Kangjie Lu[3], Xinyu Xing[2],
Ruoyu Wang[1], Adam Doupé[1], Yan Shoshitaishvili[1], Tiffany Bao[1]

[1]Arizona State University
[2]Northwestern University
[3]University of Minnesota

Linux Kernel Security

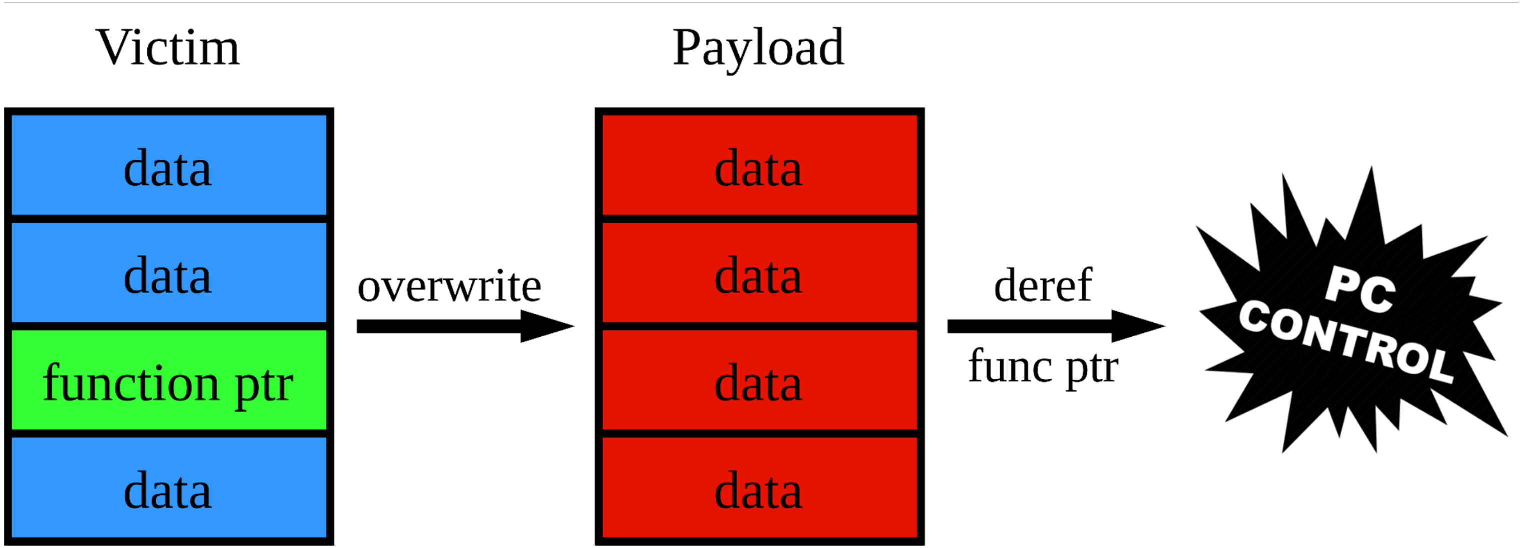
Google launched kCTF program to collect Linux kernel exploits

The maximum reward for each submission is $130,000

15 out of 16 are heap-based control-flow hijacking exploits*

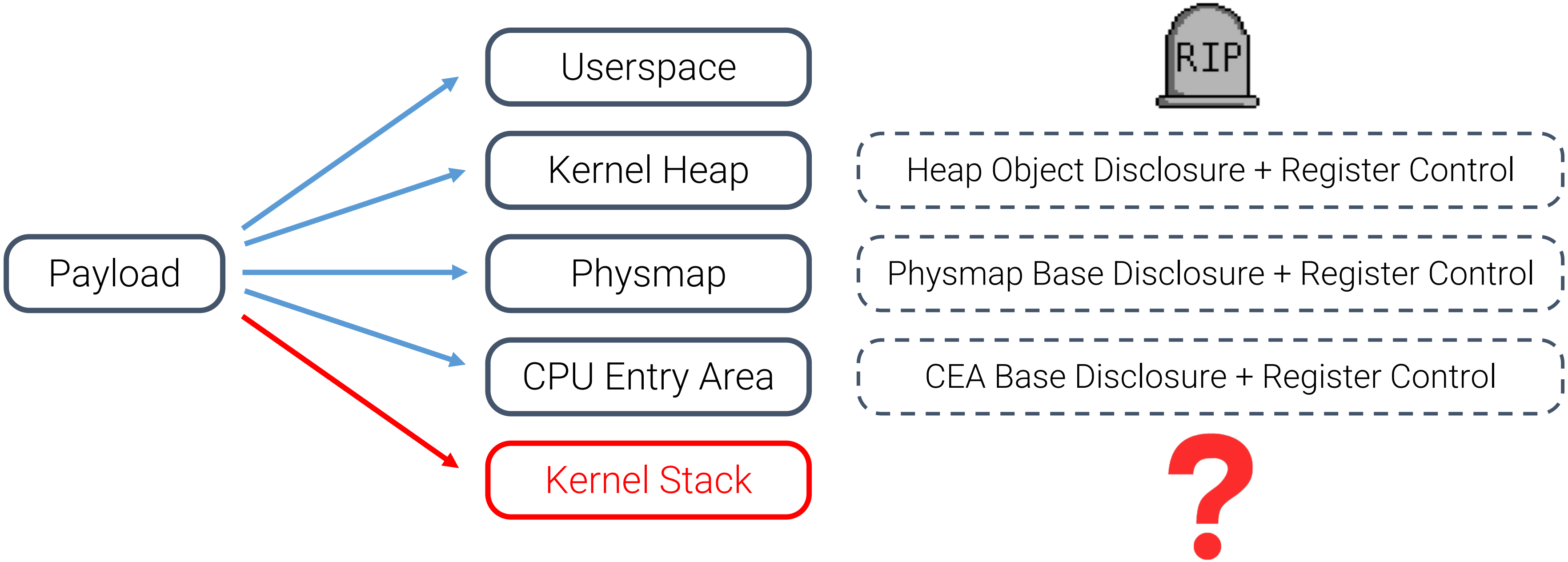
* According to Kernel Exploit Recipes Notebook by Aug 10th, 2022

Linux Kernel Heap Exploit



* According to Kernel Exploit Recipes Notebook by Aug 10th, 2022

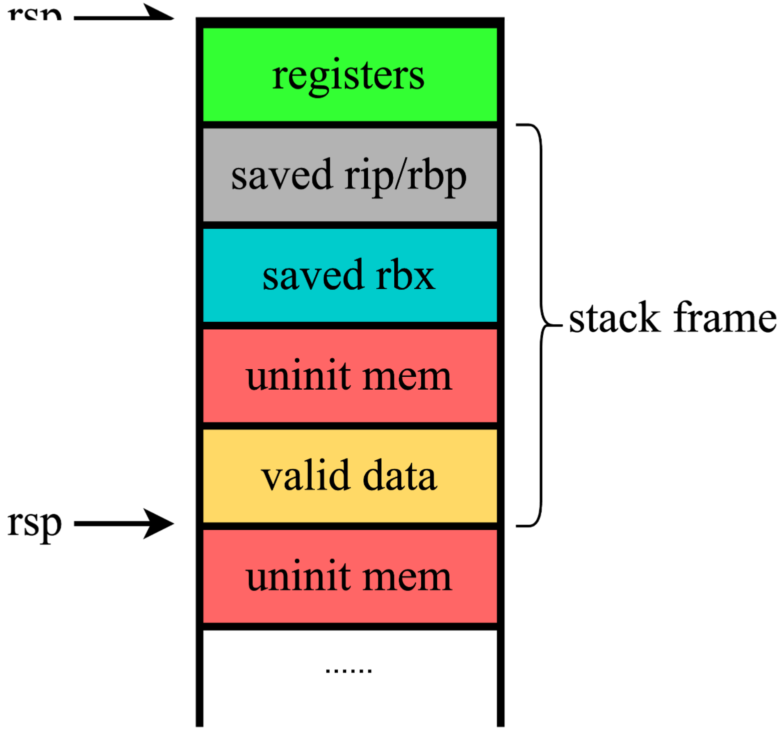
PC-CONTROL != ROOT


RIP
Payload
Userspace
Kernel Heap
Physmap
CPU Entry Area
Kernel Stack
Heap Object Disclosure + Register Control
Physmap Base Disclosure + Register Control
CEA Base Disclosure + Register Control
?

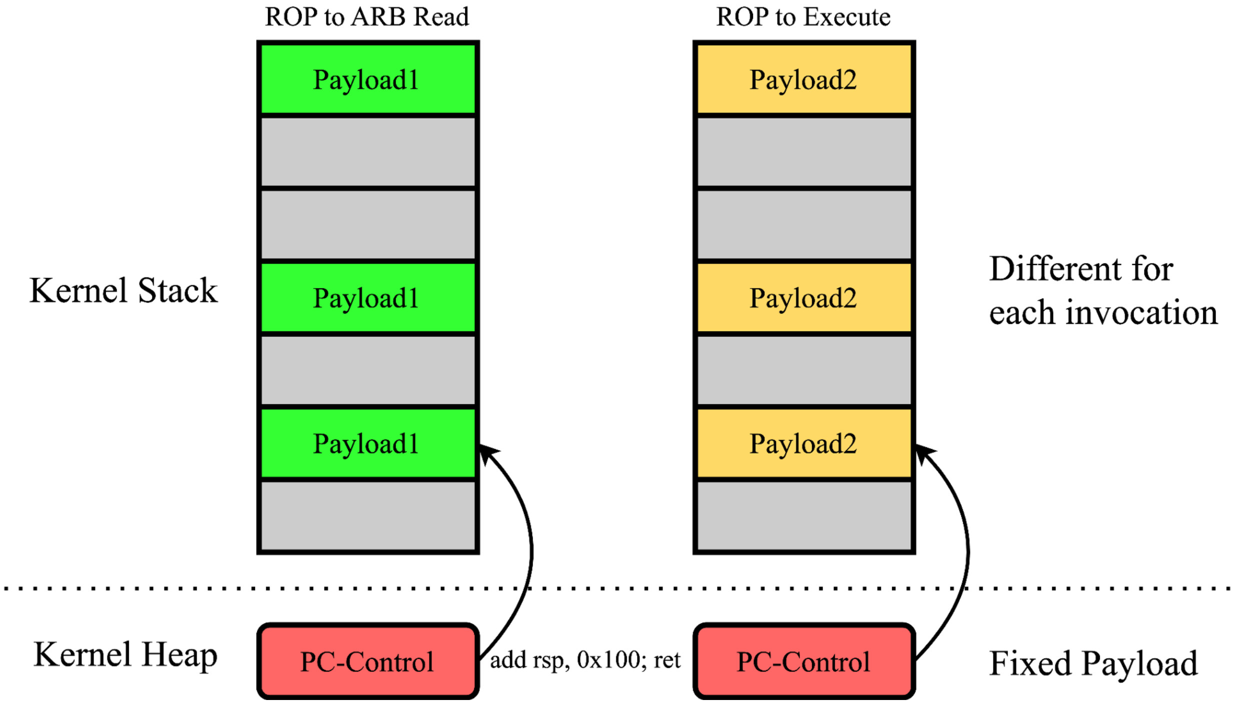
Systematically study the impact of on-stack user data on kernel security

Data Spillage Source

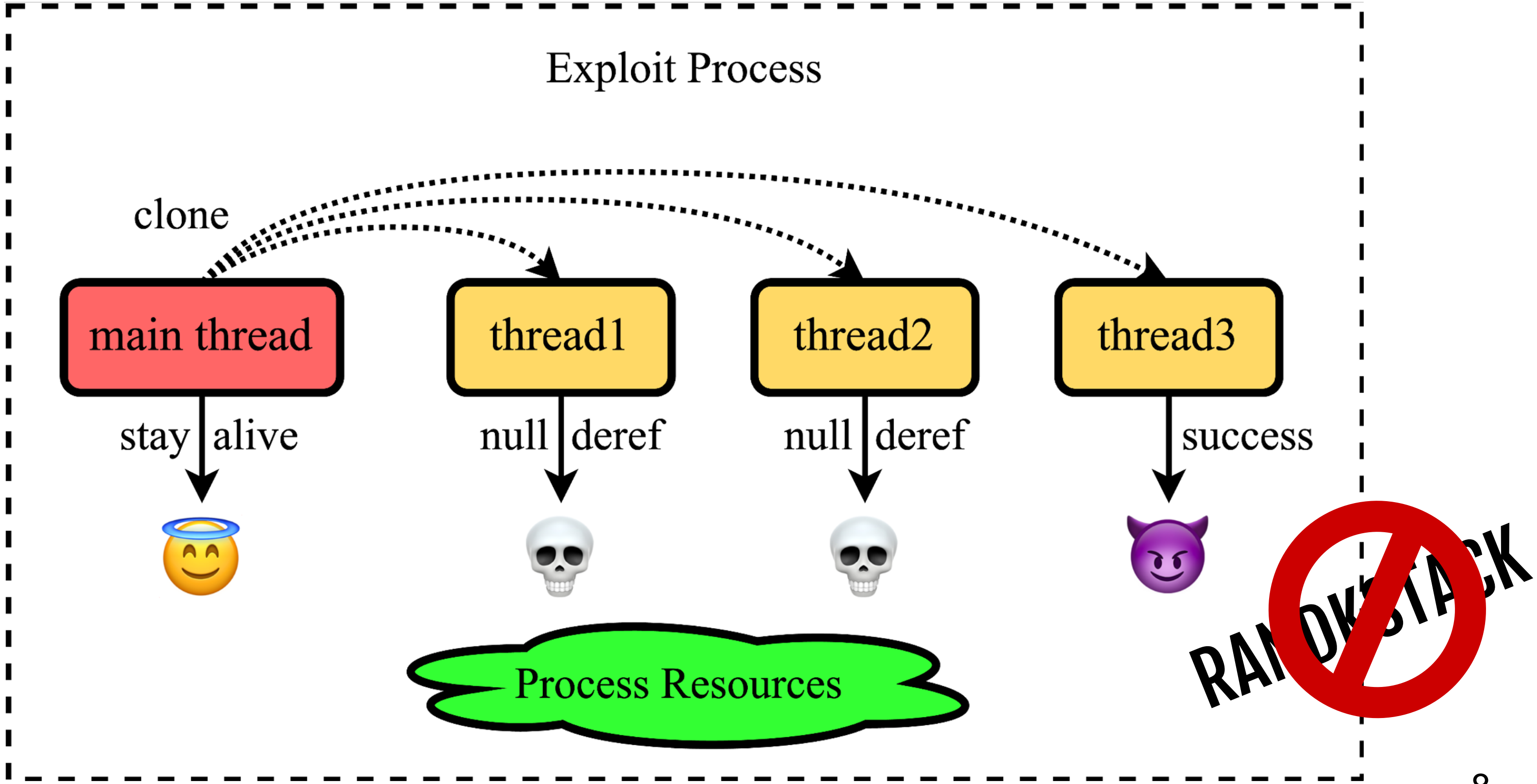
- Preserved Registers
- Calling Convention
- Valid Data
- Uninitialized Memory



Primitive 1: Rewritable Payload



Primitive 2: Crash-Resilient ROP

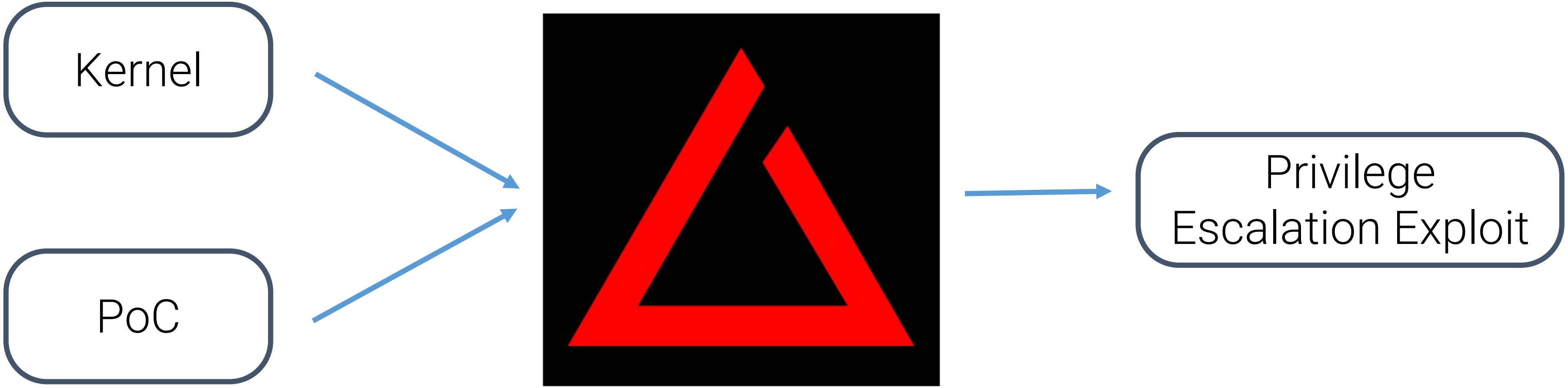


Break User/Kernel Boundary

- Rewritable Payload
  - turn one PC-Control into many without reliability degradation
- Crash-Resilient ROP
  - enhanced resiliency

RetSpill: Reliable unlimited arbitrary read/write/exec given one PC-Control

IGNI: Break User/Kernel Boundary Automatically



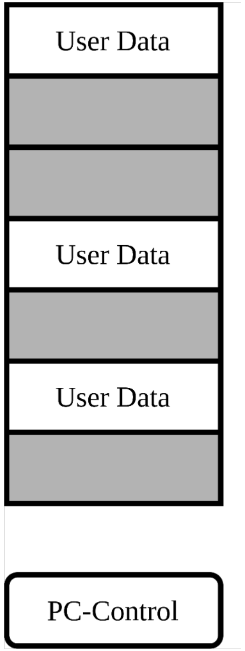
Kernel

PoC

Privilege Escalation Exploit

IGNI’s high-level workflow

IGNI: Break User/Kernel Boundary Automatically



IGNI’s high-level workflow

IGNI: Break User/Kernel Boundary Automatically

Turn ***20/22*** PoC to exploits ***automatically***

	Valid Data	Preserved Registers	Calling Convention	Uninitialized Memory	Total
Gadget	1.1	6.1	3.9	5.5	16.5

# of on-stack userspace data

RetSpill vs Mitigations

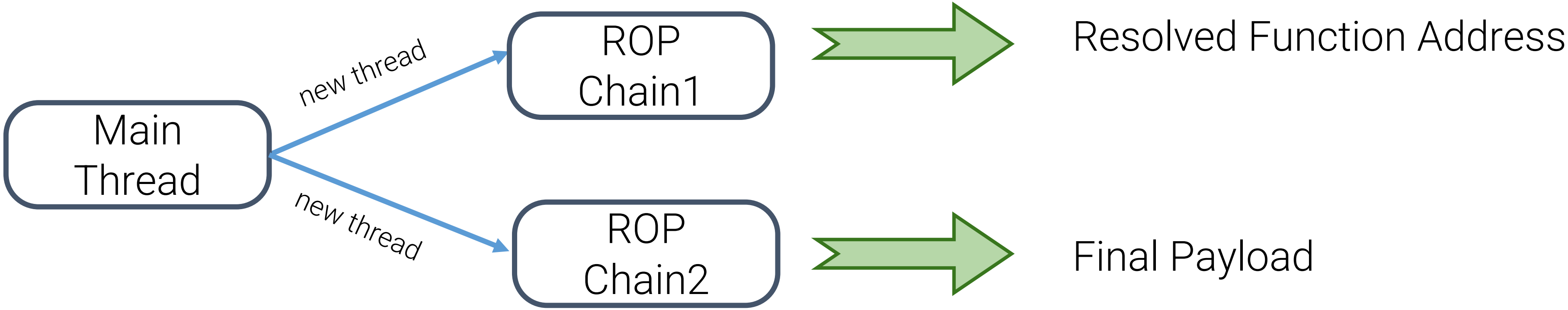
Mitigation	PC-Control Achievable?	RetSpill Works?	Deployed?
SMEP/SMAP/KPTI	✓	✓	✓
RANDKSTACK	✓	✓	✓
STACKLEAK	✓	✓	✗
FG-KASLR	✓	✓	✗
KCFI/IBT	✓	✓	✗
Shadow Stack	✓	✓?	✗
CFI+Shadow Stack	✗	✗	✗

Case Study: FG-KASLR Bypass

FG-KASLR: Function-Granular KASLR

Function-Granular: ROP gadgets available

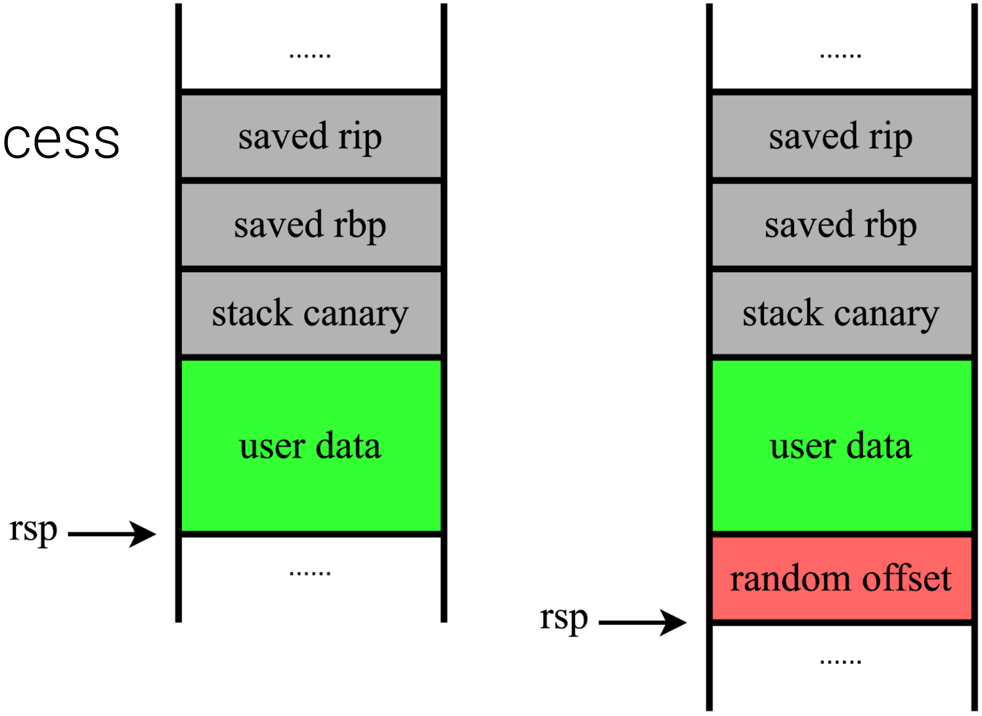
Authors of FG-KASLR updated its design after our report



Proposed Mitigation

Goal: Prevent deterministic access to any spillage data sources

Overhead: 0.61%



per-stackframe randomization

Conclusion

- Discover the RetSpill exploitation technique
- Systematically study RetSpill and demonstrate its severity
- Demonstrate the ease of exploitation with IGNI
- Propose a defense against RetSpill

RetSpill: Igniting User-Controlled Data to Burn Away Linux Kernel Protections

Thank you!

Q & A

https://github.com/sefcom/RetSpill

Kyle Zeng

✉ zengyhkyle@asu.edu

𝕏 @ky1ebot

GitHub @Kyle-Kyle