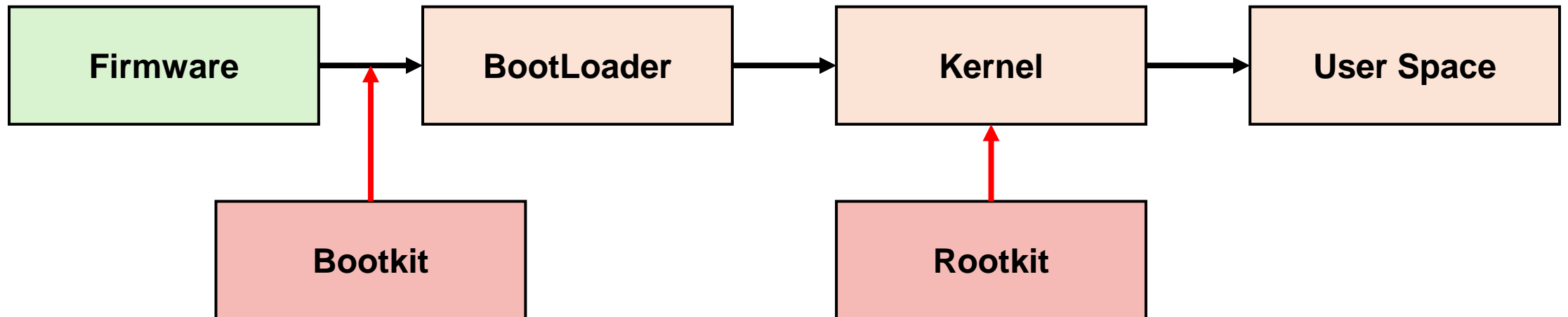


BOOTKITTY: A Stealthy Bootkit-Rootkit Against Modern Operating Systems

Junho Lee, Jihoon Kwon, HyunA Seo,
Myeongyeol Lee, Hyungyu Seo, Jinho Jung, Hyungjoon Koo

Bootkit

- **An advanced malware that hijacks a boot process**
 - Takes control of the system before the operating system launches
- **Bootkit vs. Rootkit**



Bootkits as Persistent Threats

- Bootkits have been actively studied with annual reports

ESET Research

BlackLotus UEFI bootkit: Myth confirmed

The first in-the-wild UEFI bootkit bypassing UEFI Secure Boot on fully updated UEFI systems is now a reality

LoJax UEFI Rootkit Used in Cyberespionage

October 01, 2018

Bootkitty and Linux Bootkits: We've Got You Covered

By: Paul Asadoorian | December 4, 2024

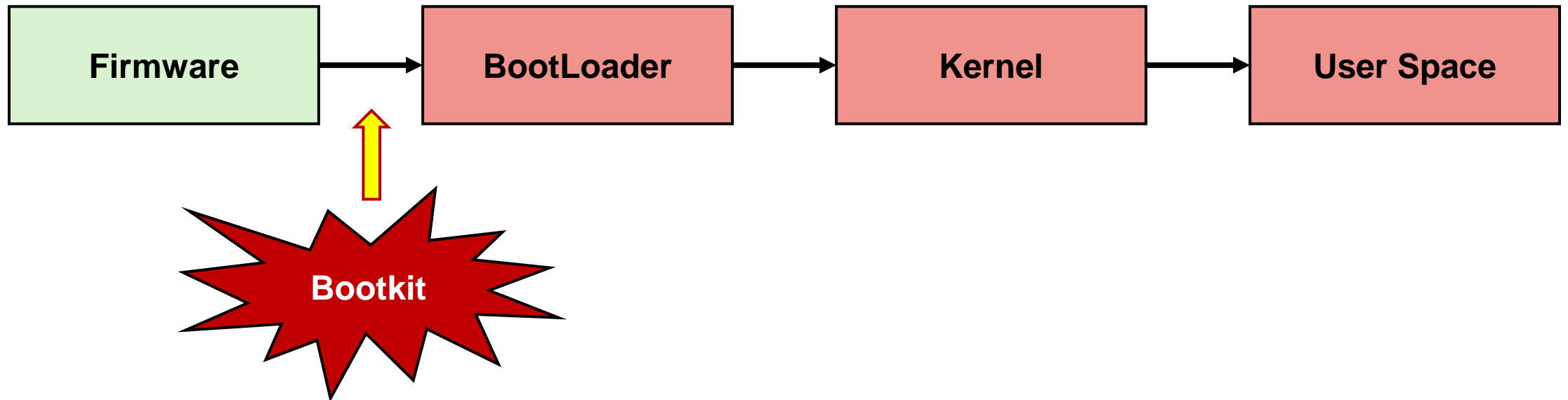
More elusive and more persistent: the third known firmware bootkit shows major advancement

January 21, 2022

Kaspersky's researchers have uncovered the third case of a firmware bootkit in the wild. Dubbed MoonBounce, this malicious implant is hidden within a computer's Unified Extensible Firmware Interface (UEFI) firmware, an essential part of computers, in the SPI flash, a storage component external to the hard drive. Such implants are notoriously difficult to remove and are of limited visibility to security products. Having first appeared in the wild in the spring of 2021, MoonBounce demonstrates a sophisticated attack flow, with evident advancement in comparison to formerly reported UEFI firmware bootkits. Kaspersky researchers have attributed the attack with considerable confidence to the well-known advanced persistent threat (APT) actor APT41.

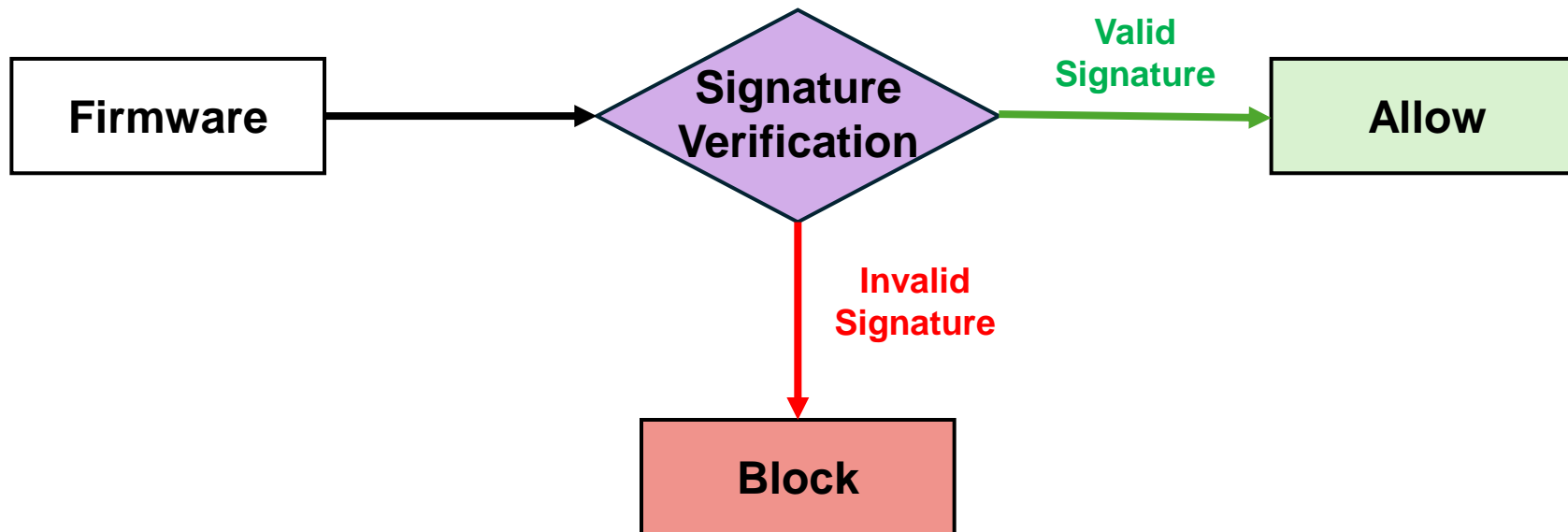
Why Dangerous?

- **Stealth:** Attacks during pre-booting, bypassing anti-viruses or EDRs
- **Persistence:** Implants into a boot partition or firmware
- **Compromise:** Controls the system before kernel initialization



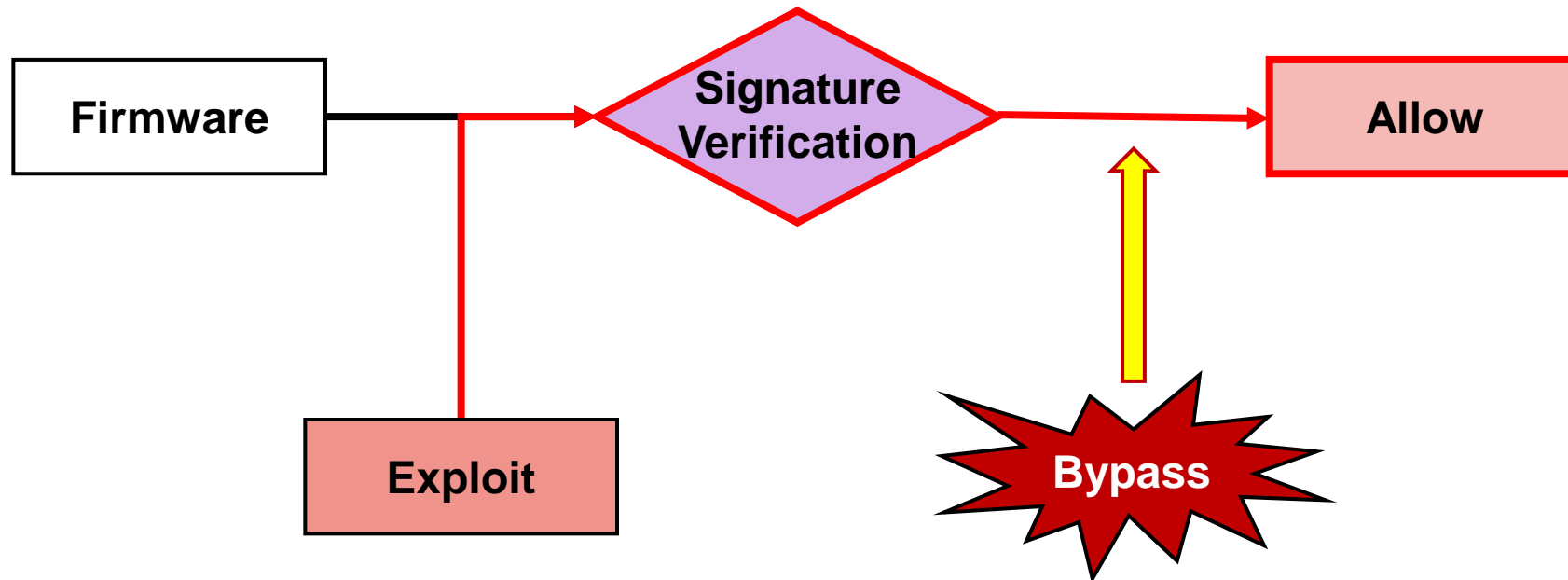
Existing Defense: Secure Boot

- Signature verification: UEFI verifies a bootloader and kernel signatures
→ **blocks untrusted code**



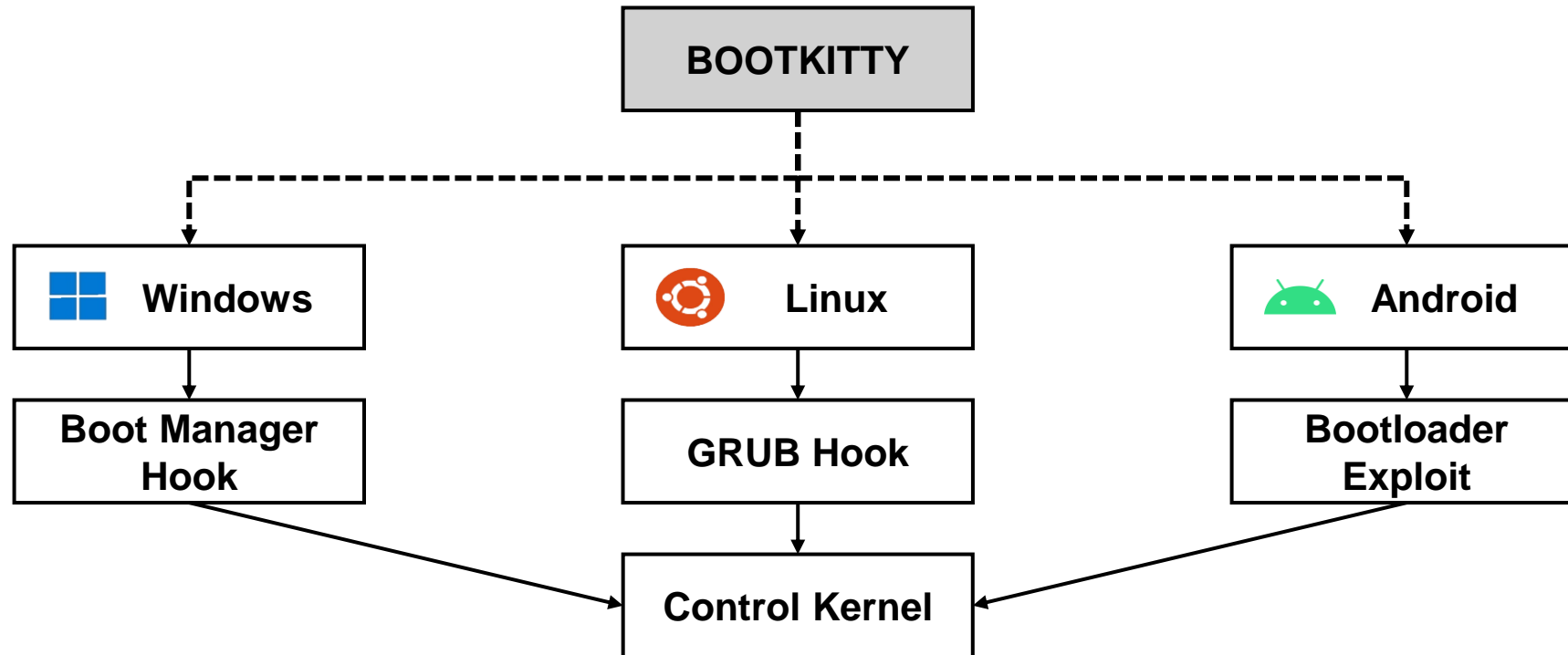
Limitations of Secure Boot: Not Fully Secure

- Secure boot has vulnerabilities in firmware and certificate management
- These vulnerabilities can be leveraged to bypass Secure Boot

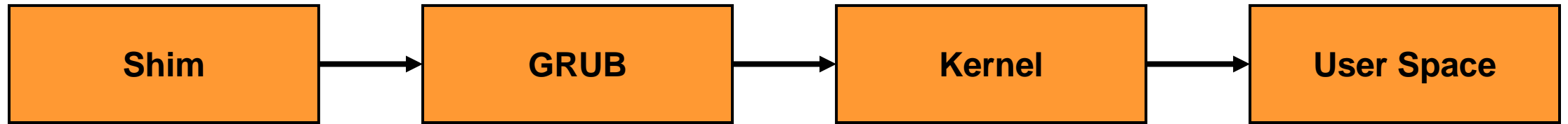


BOOTKITTY: Hybrid Bootkit-Rootkit

- BOOTKITTY hijacks the boot process and injects a stealth rootkit

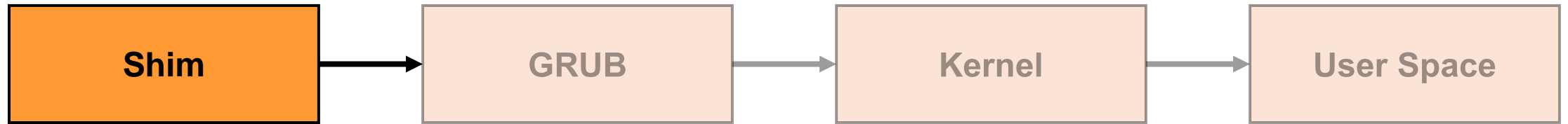


Linux Boot Process



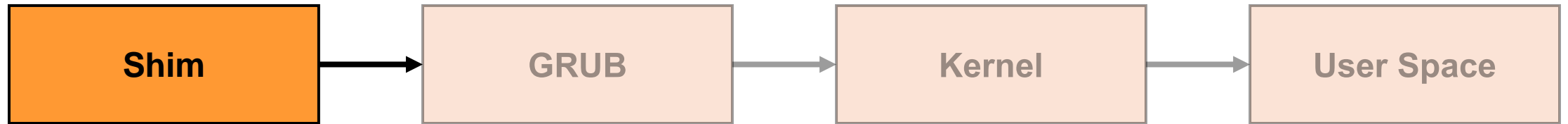
1. UEFI loads Shim
2. Shim loads GRUB
3. GRUB loads the Kernel
4. Kernel initializes the user space

Attack Surface: Linux Shim



- Verifies a bootloader signature
- Loads GRUB

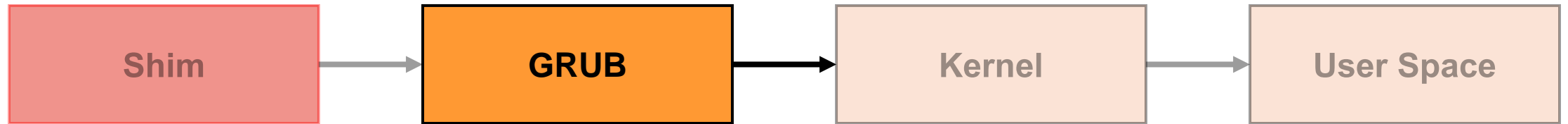
Attack Surface: Linux Shim



Attack Methods

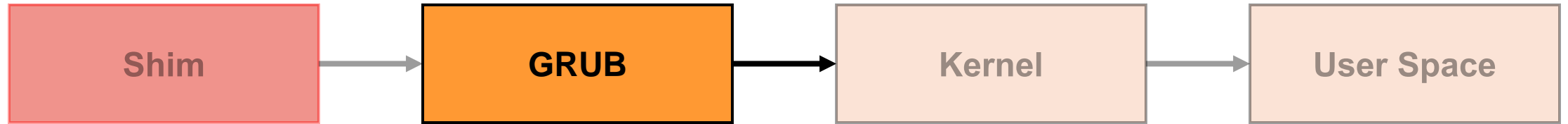
- Shim binary tampering
- Runtime hooking

Attack Surface: Linux GRUB



- Loads the kernel
- Transfers control to the kernel

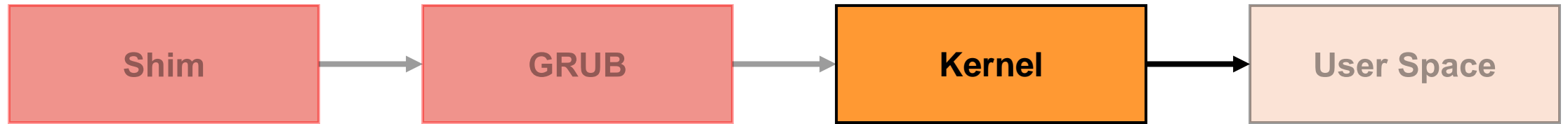
Attack Surface: Linux GRUB



Attack Methods

- GRUB modification and hooking
- Chaining hooks from Shim → GRUB → Kernel

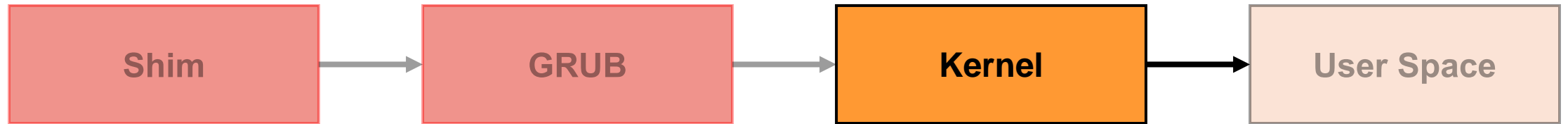
Attack Surface: Linux Kernel



Kernel loading

- Compresses the kernel
- Decompresses the kernel

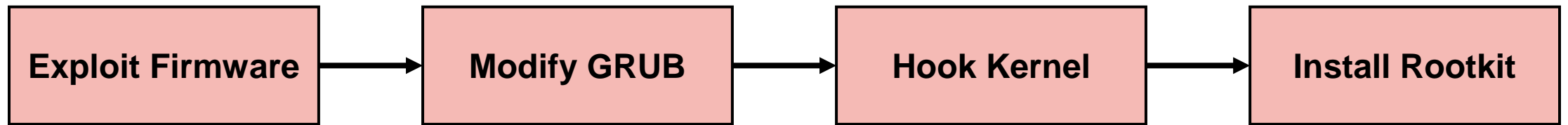
Attack Surface: Linux Kernel



Attack Methods

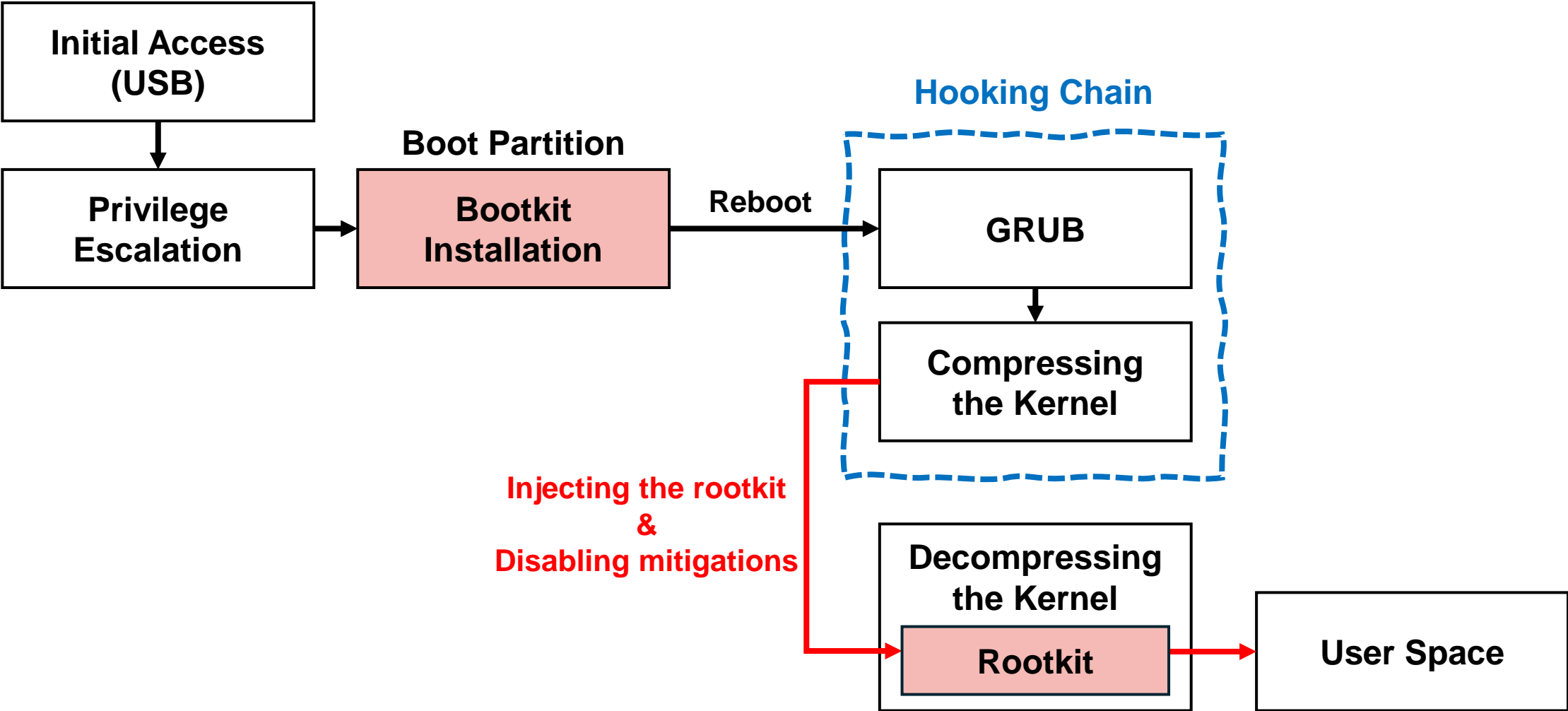
- **Before the decompression phase:** Hooks a decompression function
- **After the decompression phase:** Disable mitigations (lockdown, kernel module signing)

How BOOTKITTY Compromised the Boot Process on Linux

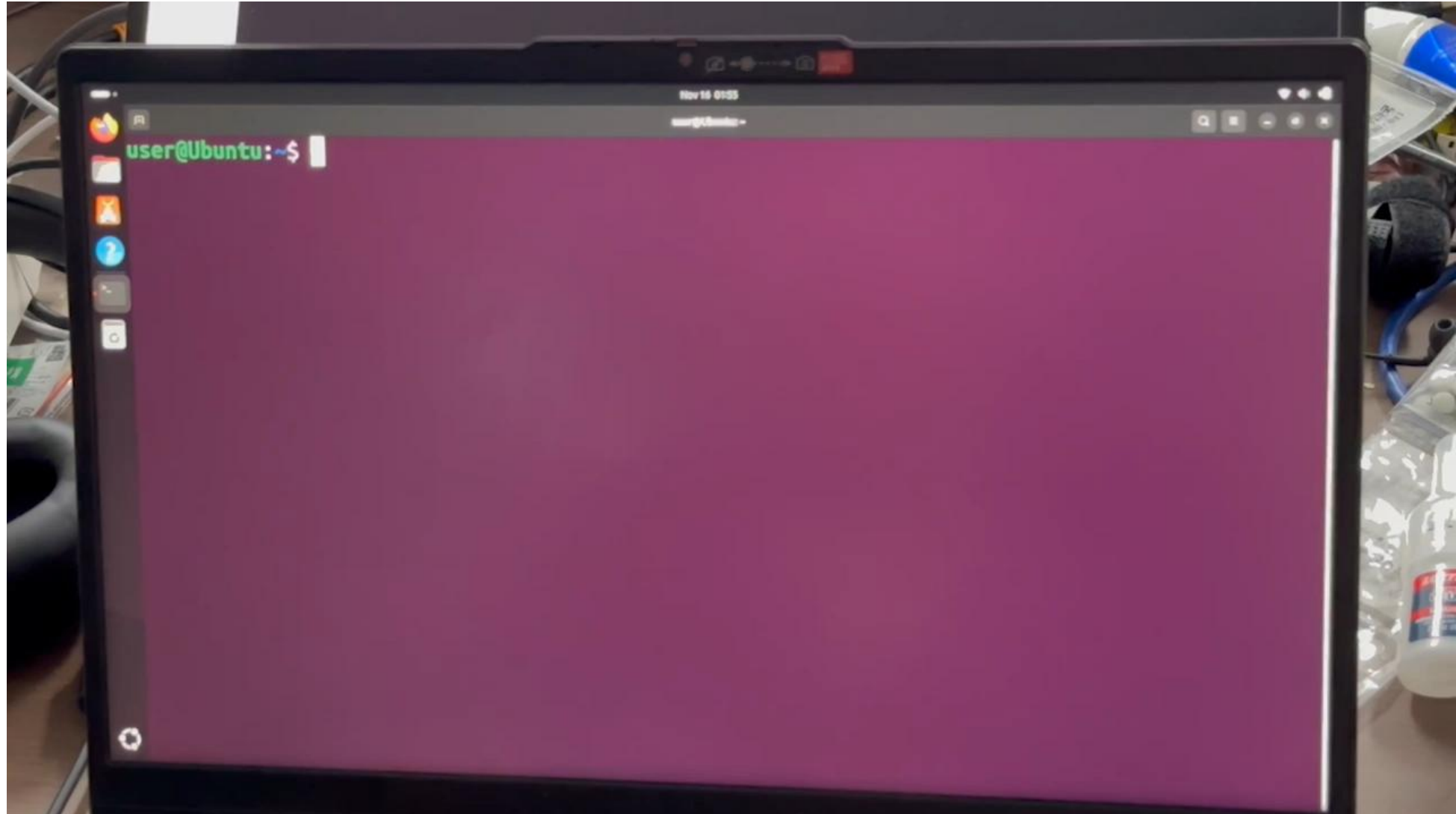


- LogoFAIL (**CVE-2023-40238**) → GRUB compromise
- GRUB modification via the firmware bug
- Chains hijacking: GRUB → kernel

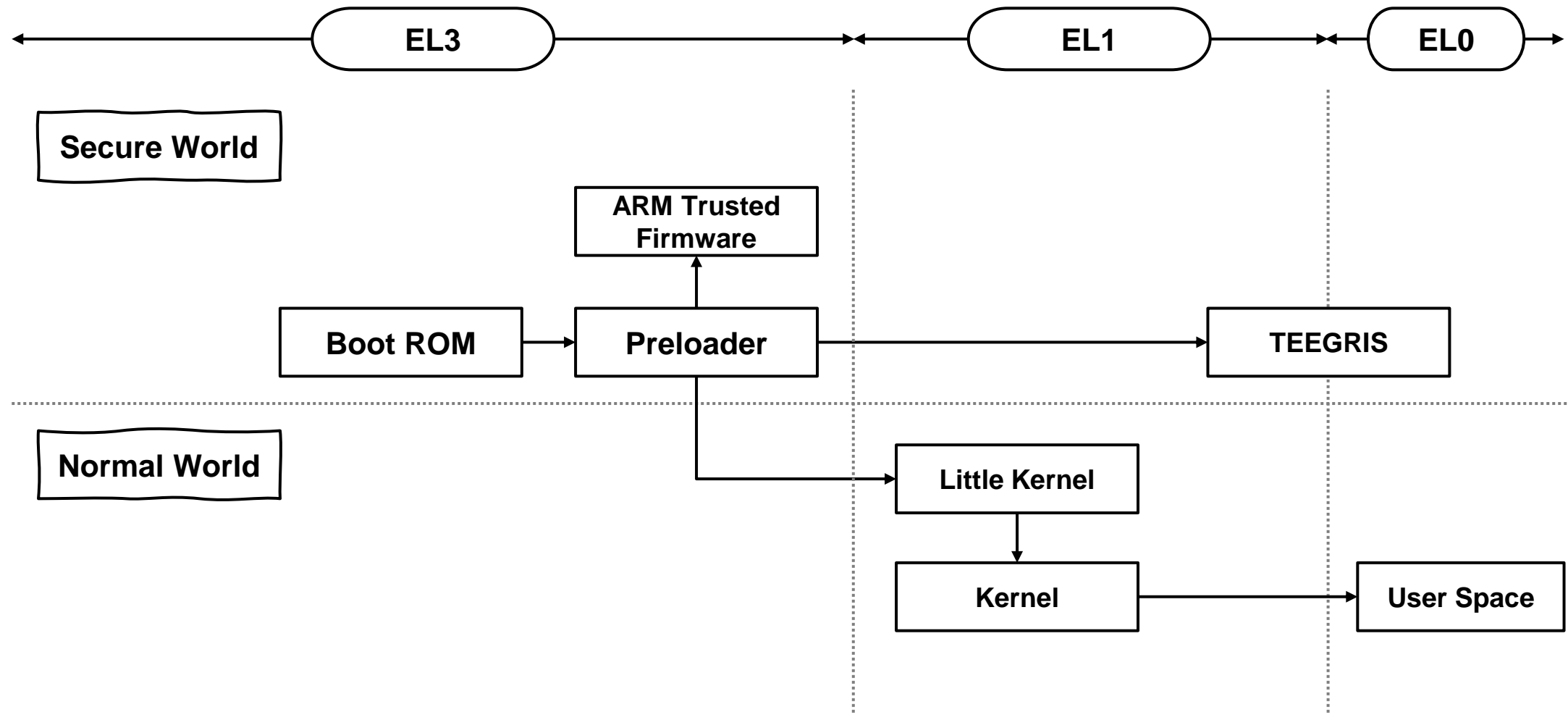
BOOTKITTY on Linux: USB Attack Scenario



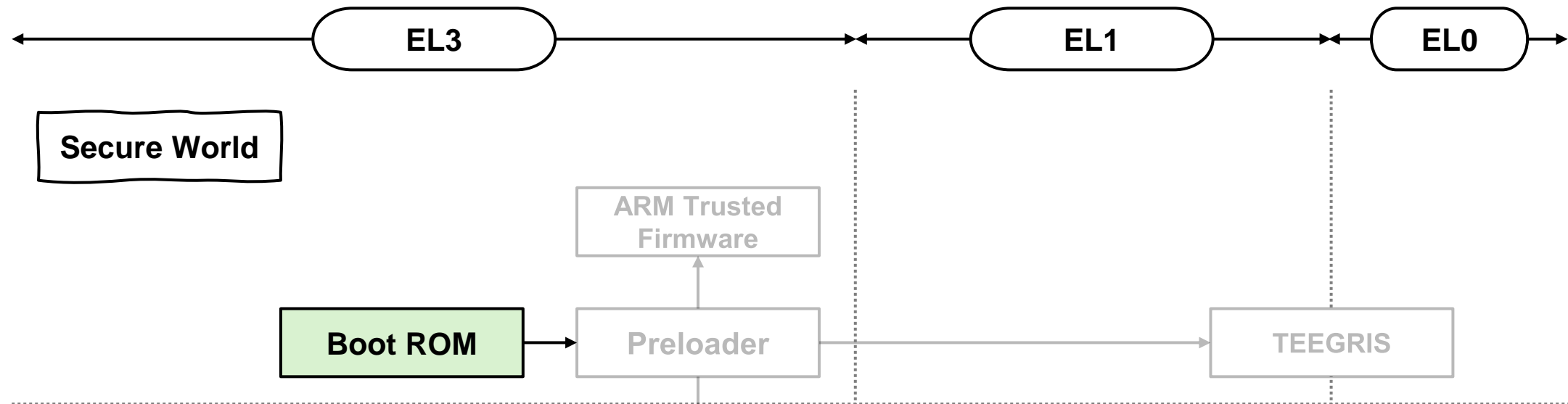
Linux Demo: Bootkit-Only



Android Boot Process: Exception Levels



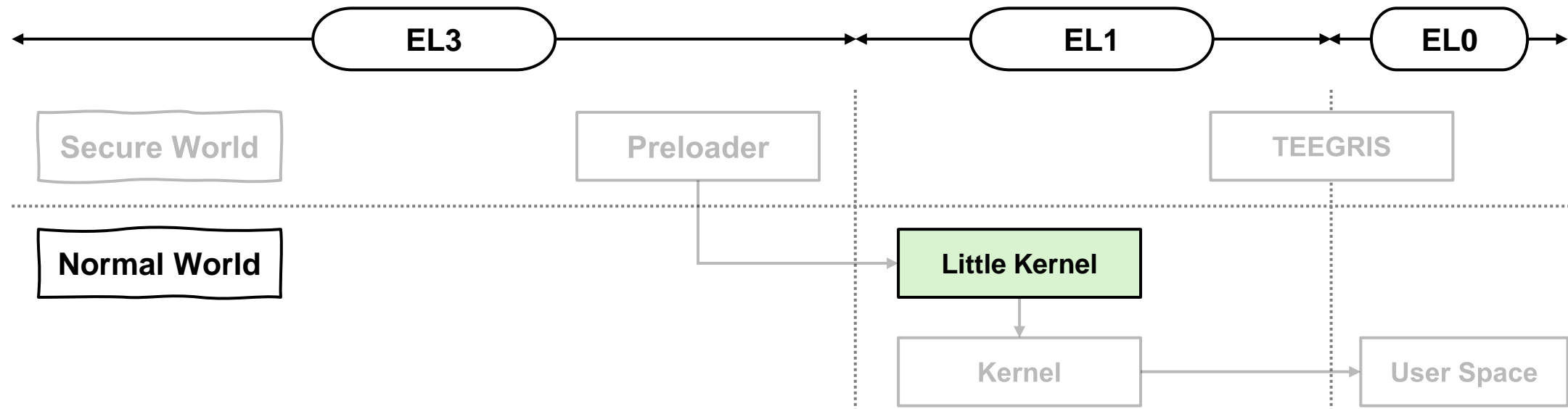
Attack Surface: Boot ROM (EL3)



Limitations

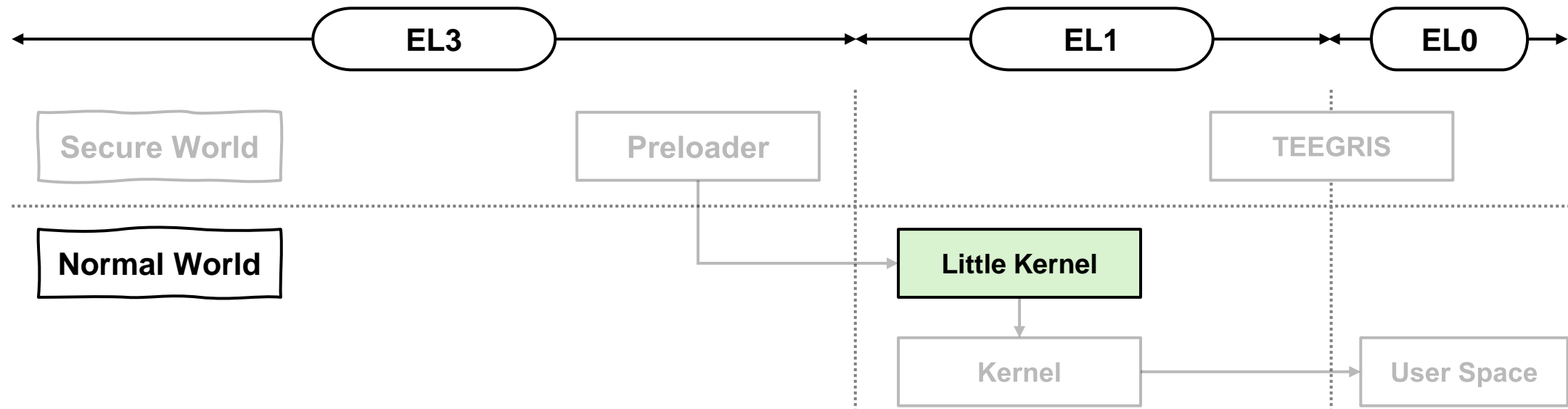
- Difficult to hook or infect
- Hard to access with software attacks

Attack Surface: Little Kernel (EL1)



- Verifies a kernel signature
- Loads the kernel
- Transfers control to the kernel

Attack Surface: Little Kernel (EL1)

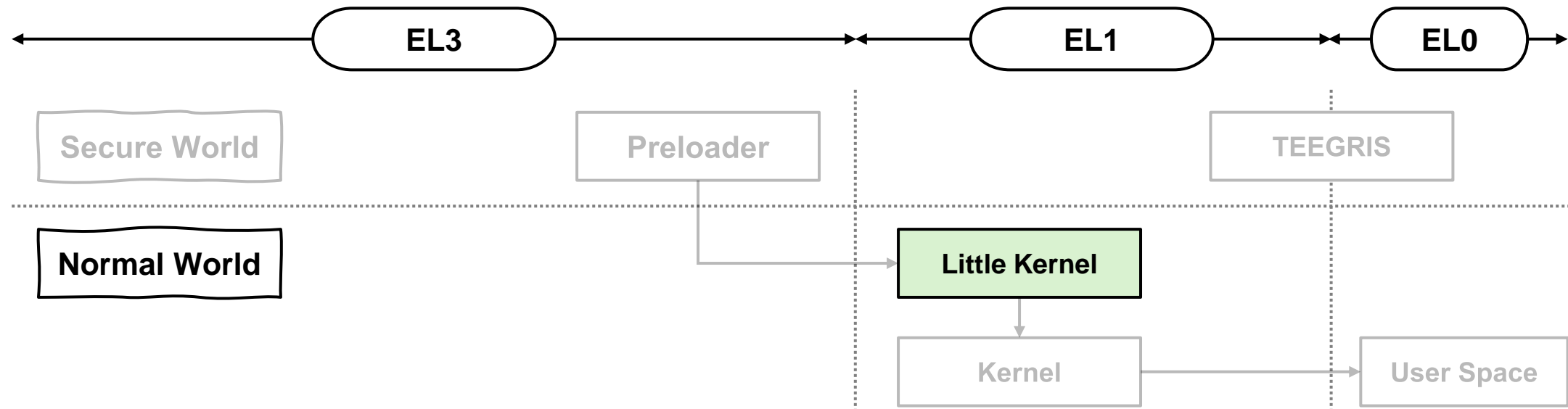


Attack Methods

- Exploits bootloader vulnerabilities

→ **Arbitrary code execution**

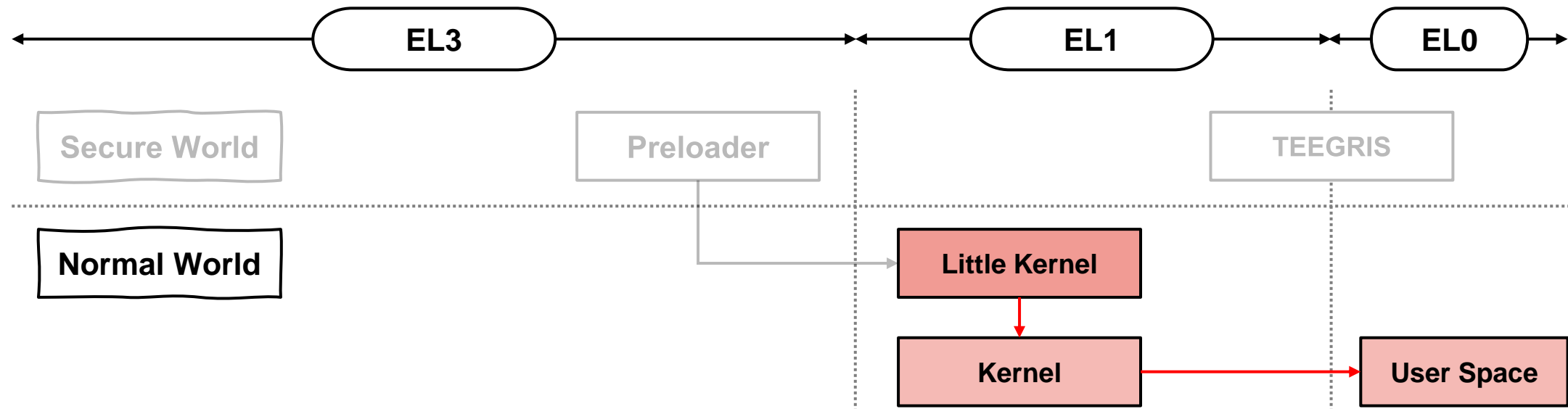
Attack Surface: Little Kernel (EL1)



Attack Methods

- Injects a hook at the kernel load point
 - **Disabling mitigations**
 - **Deploying the rootkit**

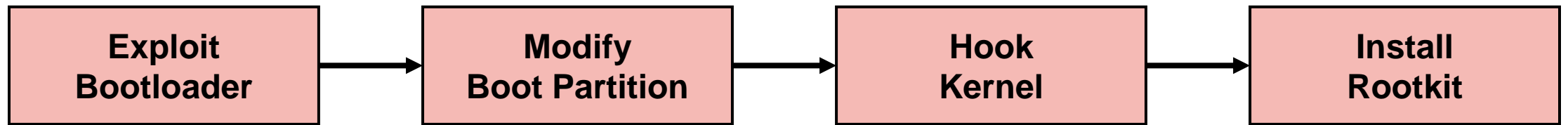
Attack Surface: Kernel (EL1 ~ EL0)



Attack Methods

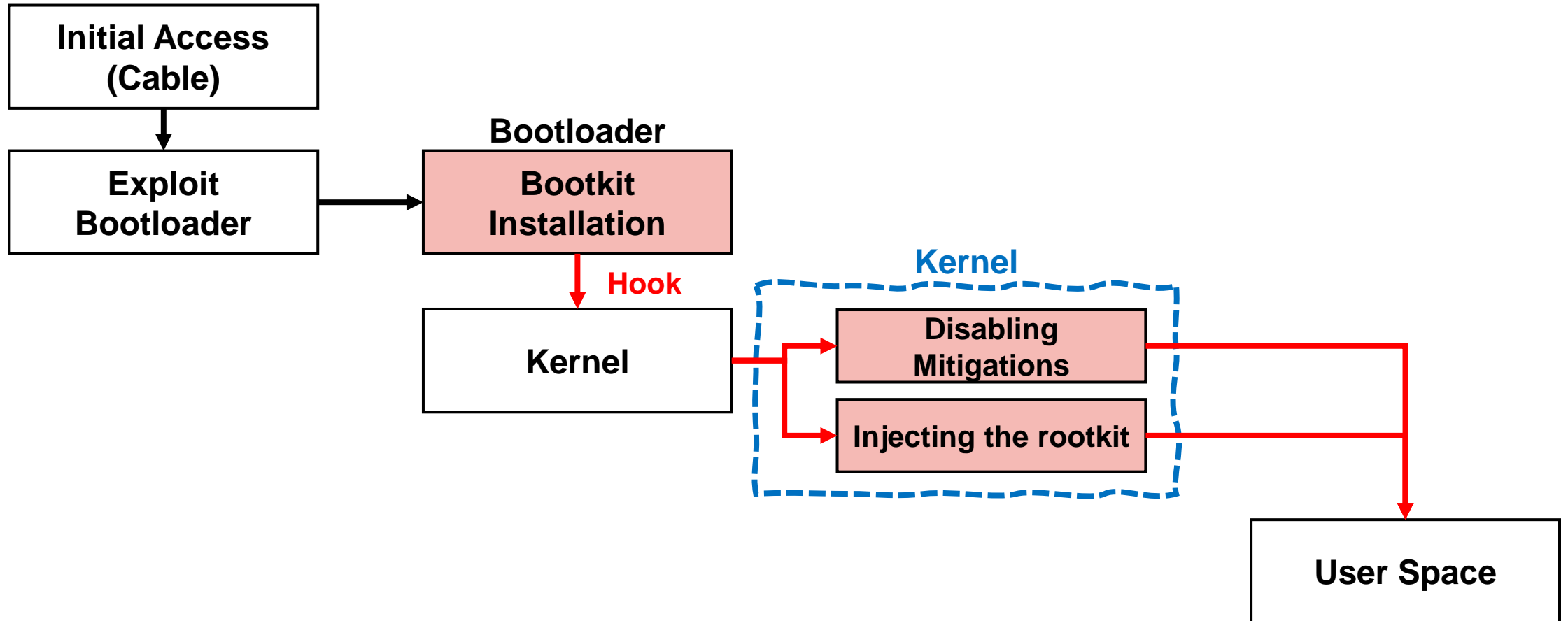
- **Before kernel execution:** Exploits bootloader vulnerability to disable mitigations (Real-time Kernel Protection, SELinux)
- **After kernel loading:** Injects and executes a rootkit before applying security policies

How BOOTKITTY Compromised the Boot Process on Android

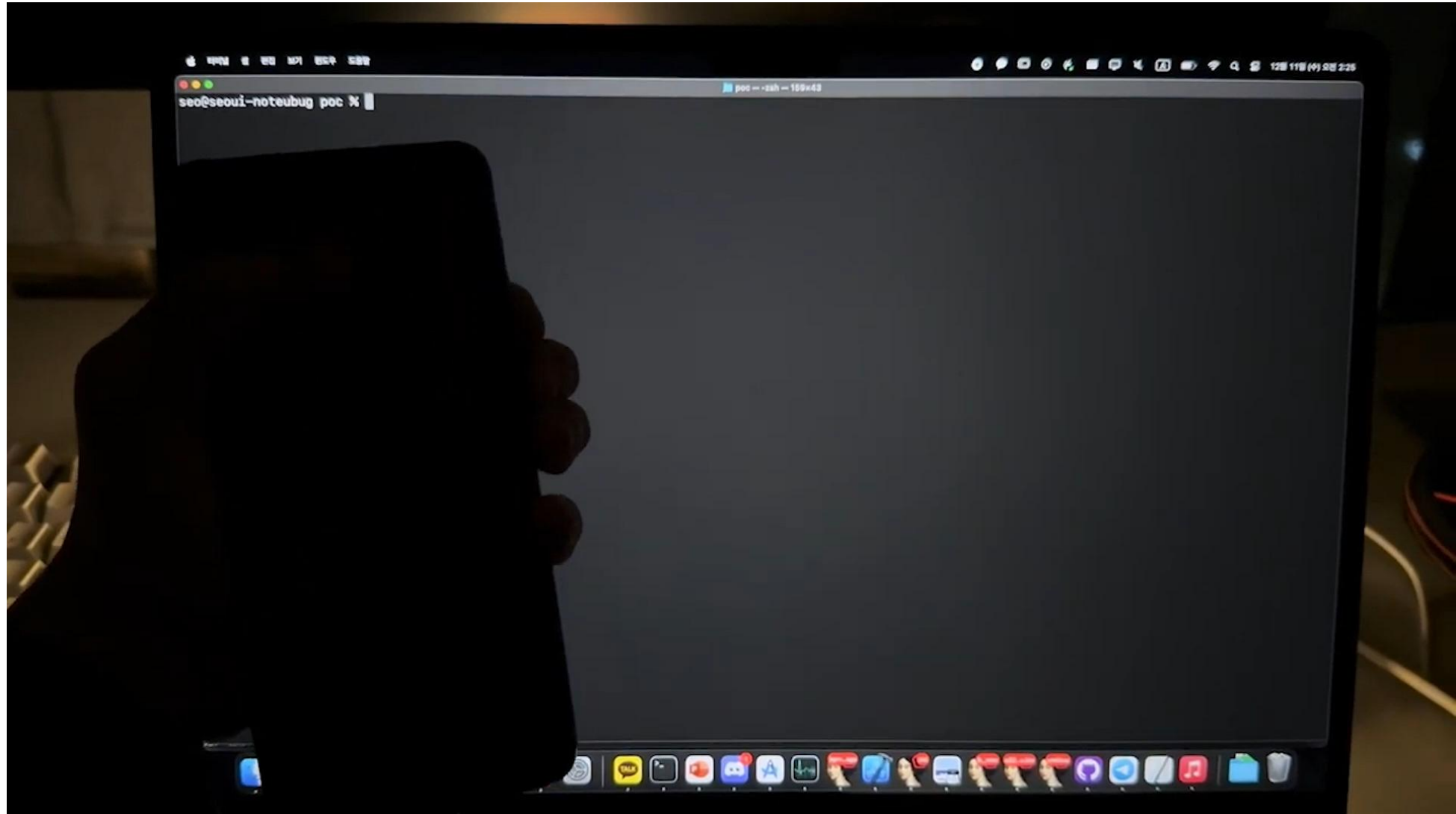


- Bootloader vulnerabilities (***CVE-2024-20832***, ***CVE-2024-20865***)
→ Boot partition compromise
- Bootkit installation via bootloader bugs
- Full-chain hijacking: Bootloader → Kernel

BOOTKITTY on Android: Bootloader Attack Scenario



Android Demo: Bootkit-Only



Key Takeaways

- Pre-kernel hooking → full system compromise
- *Hardening the boot process and boot-chain integrity verification are essential*

What We Have Not Talked About

- BOOTKITTY on windows
 - Windows boot process structure
 - Attack surfaces within the Windows boot process
 - Bootkit implementation on windows
- OS-specific techniques and bypass methods
- Rootkit deployment flow

Please read our paper!

Artifacts

- VM link



(<https://zenodo.org/records/15501870>)

- QEMU link



(<https://zenodo.org/records/15582744>)