## Diving Into Linux Kernel Security

### Alexander Popov

positive technologies



December 14-15, 2024

• Operating system security seems like a very complex topic

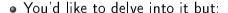
- Operating system security seems like a very complex topic
- A lot of research articles are published about it

- Operating system security seems like a very complex topic
- A lot of research articles are published about it
- You'd like to delve into it but:
  - You don't know where to start
  - You are afraid to get lost in huge amount of materials

- Operating system security seems like a very complex topic
- A lot of research articles are published about it
- You'd like to delve into it but:
  - You don't know where to start
  - You are afraid to get lost in huge amount of materials
- You may also need to configure the security parameters of your Linux systems

- Operating system security seems like a very complex topic
- A lot of research articles are published about it
- You'd like to delve into it but:
  - You don't know where to start
  - You are afraid to get lost in huge amount of materials
- You may also need to configure the security parameters of your Linux systems
- If so, then this talk is for you

- Operating system security seems like a very complex topic
- A lot of research articles are published about it



- You don't know where to start
- You are afraid to get lost in huge amount of materials



- You may also need to configure the security parameters of your Linux systems
- If so, then this talk is for you

#### Who Am I

- Alexander Popov
- Linux kernel developer since 2012
- Maintainer of some free software projects
- Principal security researcher and Head of
  - Open Source Program Office at **positive technologies**
- Conference speaker:

OffensiveCon, Nullcon Goa, Linux Security Summit, Still Hacking Anyway, Zer0Con, HITB,

Positive Hack Days, ZeroNights, HighLoad++, Open Source Summit, OS Day, Linux Plumbers

a13xp0p0v.github.io/conference\_talks





## Agenda

- Basic terminology
- Approach to learning operating system security area
- Overview of Linux kernel security
- The tool for checking Linux security parameters
- Goal: to interest you
- Bonus: invite you to join open source projects
   (this talk is about my personal OSS projects)

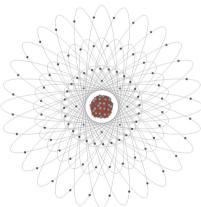


#### Disclaimer



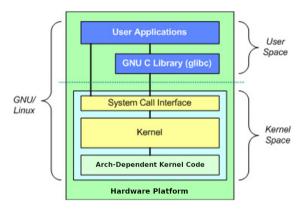
# Set Up the Terminology: OS

- An operating system is software that
  - Manages HW and SW resources
  - Provides services for computer programs
- An OS kernel is a part of the OS that runs in the privileged CPU mode
- The kernel manages processes and their usage of
  - CPU time
  - Memory and storage
  - I/O devices
  - IPC



pediaa.com/difference-between-uranium-and-thorium

# OS Architecture (GNU/Linux)



 $\underline{\mathsf{ibm.com/devel}\,\mathsf{operworks/linux/library/l-linux-kernel/}}$ 

## Set Up the Terminology: Security Model

- To build security, you first need a threat model
- Without a threat model, security measures are just a bunch of tricks for
  - Weakening system performance
  - Making users mad
- A threat model is required to create a security model

A security model describes

how security measures work together to mitigate the relevant threats



## Set Up the Terminology: OS Security Model

- An OS threat model may include
  - Parsing and handling untrusted data
  - Executing untrusted applications (userspace code)
  - Communicating via untrusted networks



Yefim Deshalyt: The heroic defense of Old Ryaza

# Set Up the Terminology: OS Security Model

- An OS threat model may include
  - Parsing and handling untrusted data
  - Executing untrusted applications (userspace code)
  - Communicating via untrusted networks
- These attack vectors against OS may be used for
  - Remote code execution (RCE)
  - 2 Local privilege escalation (LPE)
  - Openial of service (DoS)
  - Oata leaks



Yefim Deshalyt: The heroic defense of Old Ryaza

# Set Up the Terminology: OS Security Model

- An OS threat model may include
  - Parsing and handling untrusted data
  - Executing untrusted applications (userspace code)
  - Communicating via untrusted networks
- These attack vectors against OS may be used for
  - Remote code execution (RCE)
  - 2 Local privilege escalation (LPE)
  - Oenial of service (DoS)
  - Oata leaks



Yefim Deshalyt: The heroic defense of Old Ryaza

- An OS security model defines how OS security features mitigate these threats
- Excellent example: Android Platform Security Model

#### Main Term: Attack Surface

The **attack surface** covers all the system interfaces that an attacker can interact with





#### Main Term: Attack Surface

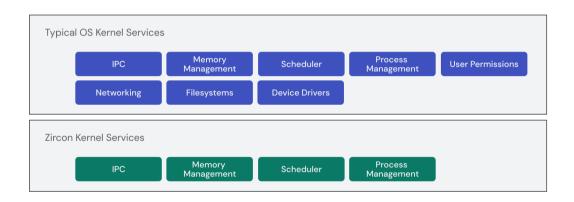
The **attack surface** covers all the system interfaces that an attacker can interact with





A bug reachable via the attack surface is a **vulnerability** 

### Comparing Linux Kernel and Zircon Microkernel



fuchsia.dev/fuchsia-src/get-started/learn/intro/zircon?hl=en

## Linux Kernel: Huge Codebase

- The Linux kernel is being developed at incredible speed
- Development statistics:
  - More than 26 million lines of code (v6.11)
  - A new major release every 9 or 10 weeks
  - Around 2000 developers contribute to the kernel every release
  - Around 14000 changesets are merged into the mainline on every release



# Fixing Bugs in the Linux Kernel

- We have a lot of bug detection tools:
  - Kernel sanitizers
  - Syzkaller kernel fuzzer
  - Various static analysis tools



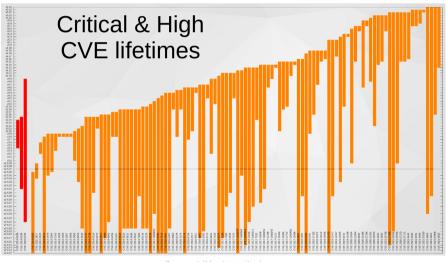
# Fixing Bugs in the Linux Kernel

- We have a lot of bug detection tools:
  - Kernel sanitizers
  - Syzkaller kernel fuzzer
  - Various static analysis tools



- However, kernel vulnerabilities appear faster than they are fixed
- Proof: A tale of a thousand kernel bugs by Dmitry Vyukov
- Moreover, kernel vulnerabilities have long lifetime

# Upstream bug lifetime – 5,5 years (Kees Cook, 2021)



outflux.net/slides/2021/lss/

### Another Approach

- To improve Linux kernel security, bug fixing alone is insufficient
- The OS kernel should handle errors safely (not giving a chance to attackers)
- grsecurity and PaX were the pioneers in this approach
- Their ideas inspired the Kernel Self Protection Project (KSPP)
- KSPP goal: kill whole bug classes and exploit methods in the mainline kernel
- KSPP overview by Kees Cook: outflux.net/slides/2021/lss/kspp.pdf



## Linux Kernel Security

Linux kernel security is a very complex knowledge area. Key concepts:

- Vulnerability classes
- Exploitation techniques
- Bug detection mechanisms
- Defence technologies
  - In the mainline
  - Shipped separately (commercial or under development)
  - Requiring special hardware features

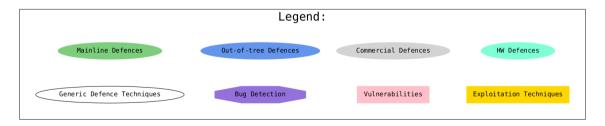


Drawn by Daniel Reeve, made by weta

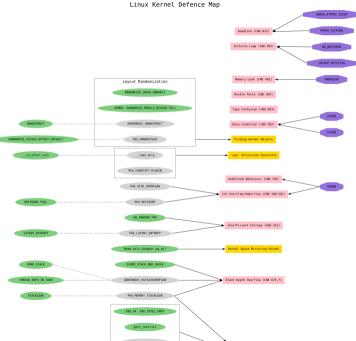
All they have complicated relations with each other...
It would be nice to have a graphical representation of it.

### Linux Kernel Defence Map

- So I developed the Linux Kernel Defence Map github.com/a13xp0p0v/linux-kernel-defence-map
- I started this project in 2018, and I'm continuing to improve and update it
- Map legend with key concepts:

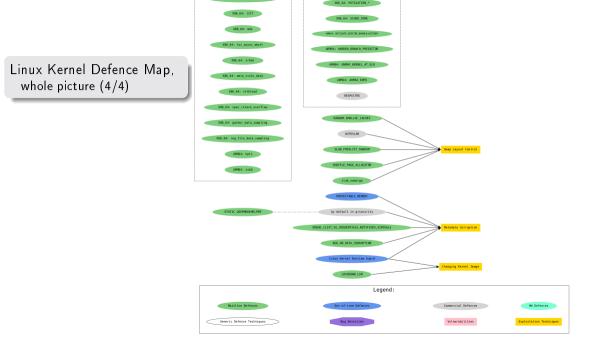


Linux Kernel Defence Map, whole picture (1/4)



GRKERNSEC HTDESYN Info Exposure (OvE-200) SECURITY DMESG RESTRICT GRKERNSEC DNESG Uninitialized Memory Usage (CNE-908) INIT STACK ALL ZERO STRUCTLEAK BYREF ALL PAY HEMDBY STRUCTLEAK init on free Linux Kernel Defence Map, PACE DOTCONTAC whole picture (2/4)Heap free() Poisoning PAY MEMORY SANTYTZE init\_on\_alloc Use-After-Free (OHE-416) slub debug-P PAX USERCOPY Stack Out-of-Bounds Access (CNE-121, 119) slub debug=Z HARDEMED DISERCORS 1964 : KASAN IN TAGS WETH ARRIVE HTE Heap Out-of-Bounds Access (ONE-122,119) KFENCE FORTIFY SOURCE Global Variable Out-of-Bounds Access (Owi-7,119) KASAN GENERIC UBSAN BOUNDS Double Free (CWE-415) SLAB\_FREELIST\_HARDENED Allocator Data Corruption by default in grsecurity bof iit harden JIT Mouse GRKERNSEC\_JIT\_HARDEN MODILE STG\* LDISC AUTOLOAD is not set Bad Module Loading request module can() GRKERNSEC MODHARDEN STRICT (KERNEL, MODULE) RICK DEBUG MX ARMS4: RODATA FULL DEFAULT ENABLED

ADMIA - DODATA FULL DEFAULT ENABLED Control Flow Hijack Techniques ARM64: ARM64 BTI KERNEL PAX KERNEXEC → ret2dir ARRES: ARRES PTR AUTH KERNEL STACKPROTECTOR Nature Address Overwrite XRS 64: XRS KERNEL TRT Linux Kernel Defence Map, Control Flow Integrity X86\_64: Intel CET ret2dir + ROP/30P/COP Coarse-grained Farward-edge CFI whole picture (3/4)X86 64: X86 SHADON STACK R0P/J0P/C0P Backward-edge CFI APRICE: STROOM\_CALL\_STRCK ret2usr + ROP/30P/COP Fire-praised Ferward-edge CFI SMEP/PXN CFI CLANG INCFID SHAPZPAN ARMS4: ARMS4 SW TTBRO PAN Userspace Data Access PAX\_UDEREF ARMS4: CPU SW DORADY PAN MULL Pointer Dereference (CMI-476) DEFAULT MMAP MEN ADDR-65536 XES 64: DILI-OR INSTEGRTEON PAGE TRACE ISOCRTICAL Transiest Execution Vulnerabilities (Off. 514) XB6 64: spectre v2 mitigations-auto-noist X86 64: spectre v2 user CPU MITIGATIONS XMS 64: spectre Mri YOU GAY MICEOCODE XSS 64: spec store bypass disable XRS 64: MITIGATION \* X86 64: 11tf X86 64: SCHED CORE X36 64: mds AMISA: HITIGATE SPECTRE BANKE HISTORY XMS 64: tax async abort ARMSA: HARDEN BRANCH PREDICTOR X86 64: srbds ARMS4: UMMAP KERNEL AT ELO 336 64: majo stale data ARMSA: ARMSA FRED



### Linux Kernel Defence Map: How It Works

- Each connection between nodes represents some kind of relationship between them
- The node connections don't always mean "full mitigation"
- The map helps with navigating documentation and Linux kernel sources

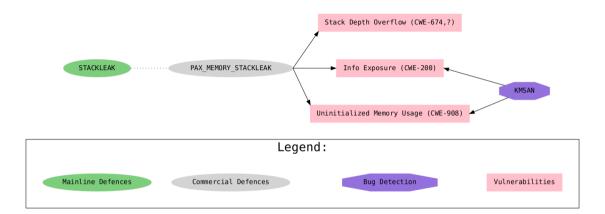
### Linux Kernel Defence Map: How It Works

- Each connection between nodes represents some kind of relationship between them
- The node connections don't always mean "full mitigation"
- The map helps with navigating documentation and Linux kernel sources
- The map provides the Common Weakness Enumeration (CWE) numbers for vuln classes

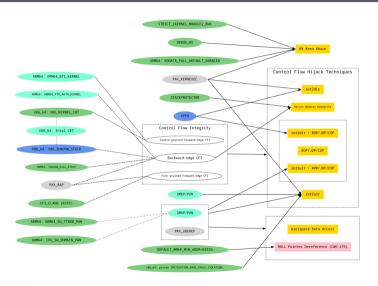
## Linux Kernel Defence Map: How It Works

- Each connection between nodes represents some kind of relationship between them
- The node connections don't always mean "full mitigation"
- The map helps with navigating documentation and Linux kernel sources
- The map provides the Common Weakness Enumeration (CWE) numbers for vuln classes
- This map describes kernel security hardening
- [!] The map doesn't cover
  - Cutting the attack surface
  - Userspace security features
  - Security policies enforced by Linux Security Modules (LSM)

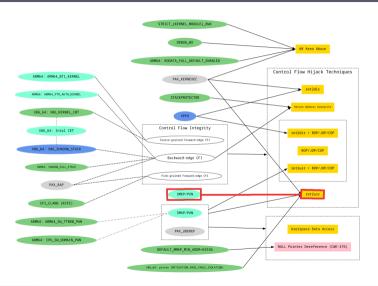
### Example from the Map: STACKLEAK



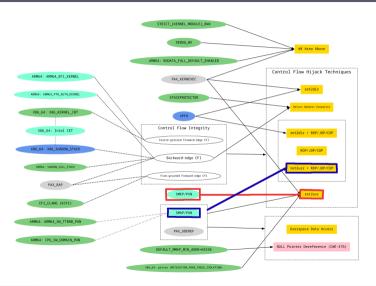
### Example from the Map: Control-Flow Hijack



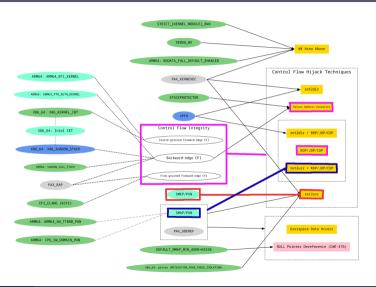
### Example from the Map: Control-Flow Hijack



## Example from the Map: Control-Flow Hijack

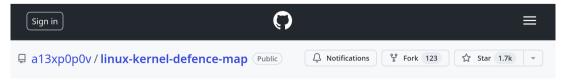


## Example from the Map: Control-Flow Hijack



### Linux Kernel Defence Map: Implementation

- The map needs to be updated (the Linux kernel is evolving)
- I want to develop it in text:
  - To use Git
  - 2 To make it a free software project under GPL-3.0
- I don't want to place nodes and edges manually
- So I use the DOT language provided by Graphviz: dot -Tsvg map.dot -o map.svg
- The project is live and successful, you're welcome to join!



# Linux Kernel Defence Map: Code Example

```
// Defences relations
edge [style=dotted, arrowhead=none, dir=none, headport= , tailport= ];
"STACKLEAK":e -> "PAX MEMORY STACKLEAK":w;
// Defences vs Vulnerabilities and Exploitation Techniques
edge [style=solid, arrowhead=normal, dir=forward, headport=_, tailport=_];
"PAX MEMORY STACKLEAK":e -> "Stack Depth Overflow (CWE-674,?)":sw;
"PAX MEMORY STACKLEAK":e -> "Uninitialized Memory Usage (CWE-908)":nw;
"PAX_MEMORY_STACKLEAK":e -> "Info Exposure (CWE-200)":w;
// Bug Detection Mechanisms vs Vulnerabilities
edge [style=solid, arrowhead=normal, dir=back, headport=_, tailport=_];
"Uninitialized Memory Usage (CWE-908)":e -> "KMSAN";
"Info Exposure (CWE-200)":e -> "KMSAN":
```

# Linux Kernel Defence Map: Knowledge Sources

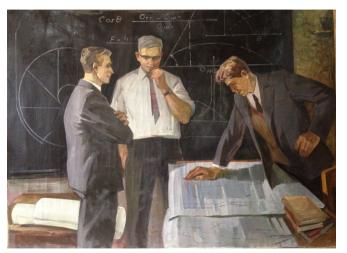
- Linux kernel security documentation
- grsecurity documentation
- Kernel Self Protection Project recommendations
- Microsoft Security Response Center (MSRC) publications



And much more:

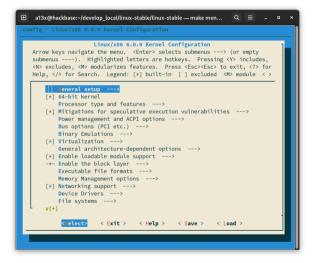
github.com/a13xp0p0v/linux-kernel-defence-map#references

# Nice Map! But What's in Practice?



Victor Belov: Soviet scientists theorists (1972)

#### In Practice We Have This!



### Linux Kernel Parameters

- Kconfig options (compile-time)
- Kernel cmdline arguments (boot-time)
- Sysctl parameters (runtime)



Buran spacecraft control panel

# Linux Kernel Security Settings

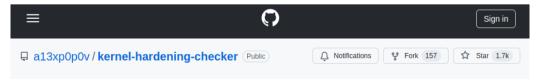
- There are plenty of Linux kernel security parameters
- A lot of them are **not enabled** by the major distros
- Nobody likes checking configs manually

# Linux Kernel Security Settings

- There are plenty of Linux kernel security parameters
- A lot of them are **not enabled** by the major distros
- Nobody likes checking configs manually
- So let computers do their job!

# Linux Kernel Security Settings

- There are plenty of Linux kernel security parameters
- A lot of them are **not enabled** by the major distros
- Nobody likes checking configs manually
- So let computers do their job!
- I created **kernel-hardening-checker** for checking the security-related parameters of the Linux kernel: github.com/a13xp0p0v/kernel-hardening-checker
- I started this work in 2018, the project is in continuous development



### kernel-hardening-checker -h

```
a || = |
 (F)
                                              travel@horizon: ~/kernel-hardening-checker
travel@horizon:~/kernel-hardening-checker$ ./bin/kernel-hardening-checker -h
usage: kernel-hardening-checker [-h] [--version] [-m {verbose.json.show ok.show fail}] [-a] [-c CONFIG] [-v KERNEL VERSION]
                                [-1 CMDLINE] [-s SYSCTL] [-p {X86 64.X86 32.ARM64.ARM}] [-q {X86 64.X86 32.ARM64.ARM}]
A tool for checking the security hardening options of the Linux kernel
ontions:
                        show this help message and exit
  -h. --help
                        show program's version number and exit
  --version
  -m {verbose ison show ok show fail}. --mode {verbose ison show ok show fail}
                        choose the report mode
                        autodetect and check the security hardening options of the running kernel
  -a. --autodetect
  -c CONFIG. --config CONFIG
                        check the security hardening options in the Kconfig file (also supports *.gz files)
  -v KERNEL VERSION. --kernel-version KERNEL VERSION
                        extract version from the kernel version file (contents of /proc/version) instead of Kconfig file
  -l CMDLINE. --cmdline CMDLINE
                        check the security hardening options in the kernel cmdline file (contents of /proc/cmdline)
  -s SYSCTL. --sysctl SYSCTL
                        check the security hardening options in the sysctl output file ('sudo sysctl -a > file')
  -D {X86 64.X86 32.ARM64.ARM}, --print {X86 64.X86 32.ARM64.ARM}
                        print the security hardening recommendations for the selected microarchitecture
  -q {X86 64,X86 32,ARM64,ARM}, --generate {X86 64,X86 32,ARM64,ARM}
                        generate a Kconfig fragment with the security hardening options for the selected microarchitecture
travel@horizon:~/kernel-hardening-checkerS
travel@horizon:~/kernel-hardening-checkerS
```

## kernel-hardening-checker: Demo Time



# kernel-hardening-checker: Under the Hood

- Free software project under GPL 3.0 license
- Written in Python
  - Please don't cry if my code looks like C code
  - I'm just a kernel developer :)
- Distribution via pip/setuptools
- Regular releases (linked to kernel releases)
- CI: automatic functional and unit tests, code coverage >97%
- Wonderful contributors from all over the world (kudos!)
- This tool is used by many GNU/Linux distributions (I'm glad!)

### kernel-hardening-checker

(formerly kconfig-hardened-check)

# kernel-hardening-checker: Ideas and Plans

Vasily Tikhonenko: Highlanders. At the new construction site (1975)

- Allow redefining rules and expanding rule sets
- Add "with care" column to mark settings that
  - May break some kernel functionality
  - Or may introduce significant performance impact
- Evaluate the performance penalty of the recommended kernel settings (depends on the workload)
- Create documentation describing Linux kernel security settings
- Add RISC-V support
- And many more: github.com/a13xp0p0v/kernel-hardening-checker/issues



### Conclusion

- The Linux Kernel Defence Map helps to:
  - Get an overview of Linux kernel security
  - Develop a threat model for your GNU/Linux system
  - Learn about kernel defences that can help against these threats



#### Conclusion

- The Linux Kernel Defence Map helps to:
  - Get an overview of Linux kernel security
  - Develop a threat model for your GNU/Linux system
  - Learn about kernel defences that can help against these threats



The kernel-hardening-checker tool helps to control the security-related parameters of your kernel

### Conclusion

- The Linux Kernel Defence Map helps to:
  - Get an overview of Linux kernel security
  - Develop a threat model for your GNU/Linux system
  - Learn about kernel defences that can help against these threats



- The kernel-hardening-checker tool helps to control the security-related parameters of your kernel
- Please don't change these settings without knowing your threat model

# Thanks! Obrigado!

# Enjoy the conference!

Contacts:

a13xp0p0v

a13xp0p0v@tuta.io

Blog: a13xp0p0v.github.io



Channel: t.me/linkersec





