STACKLEAK: A Long Way to the Linux Kernel Mainline

Alexander Popov

Positive Technologies

August 27, 2018, Vancouver

LINUX SECURITY SUMMIT

About Me

- Alexander Popov
- Linux kernel developer
- Security researcher at POSITIVE TECHNOLOGIES

Agenda

- STACKLEAK overview, credit to grsecurity/PaX
- My role
- STACKLEAK as a security feature
  - Affected kernel vulnerabilities
  - Protection mechanisms
  - Performance penalty
- The way to the Mainline
  - Timeline and the current state
  - Changes from the original version
  - Interactions with Linus and subsystem maintainers

STACKLEAK Overview

- Awesome Linux kernel security feature
- Developed by PaX Team (kudos!)
- `PAX_MEMORY_STACKLEAK` in grsecurity/PaX patch
- grsecurity/PaX patch is not freely available now
- The last public version is for 4.9 kernel (April 2017)

My Goal

Bring STACKLEAK into the Linux kernel mainline

Thanks to Positive Technologies for allowing me to spend part of my working time on it!

Thanks to my wife and kids for allowing me to spend plenty of my free time on it!

My Tactics

- Extract STACKLEAK from grsecurity/PaX patch

```
$ wc -l ../grsecurity-3.1-4.9.24-201704252333.patch
225976 ../grsecurity-3.1-4.9.24-201704252333.patch
```

- Carefully learn it bit by bit
- Send to LKML, get feedback, improve, repeat ...

My Tactics

- Extract STACKLEAK from grsecurity/PaX patch

```
$ wc -l ../grsecurity-3.1-4.9.24-201704252333.patch
225976 ../grsecurity-3.1-4.9.24-201704252333.patch
```

- Carefully learn it bit by bit
- Send to LKML, get feedback, improve, repeat ...

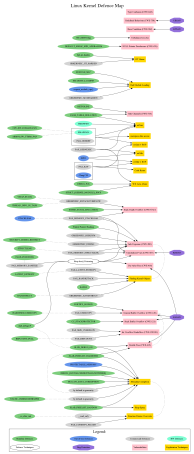
for more than a year: **15** versions of the patch series

STACKLEAK Security Features

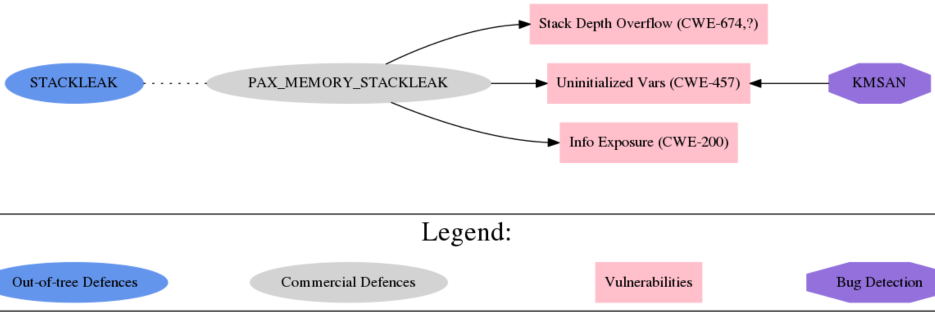
Now about STACKLEAK security features

Linux Kernel Defence Map: Whole Picture

https://github.com/a13xp0p0v/linux-kernel-defence-map



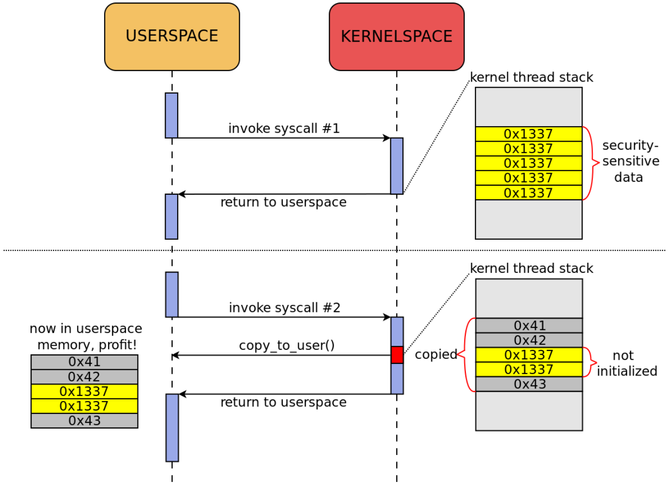
Linux Kernel Defence Map: STACKLEAK Part



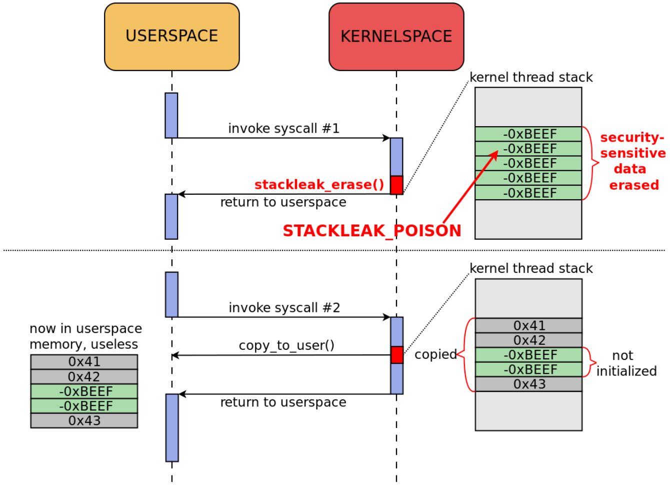
STACKLEAK Security Feature 1

- Erases the kernel stack at the end of syscalls
- Reduces the information that can be revealed through some* kernel stack leak bugs

Kernel Stack Leak Bug Example



STACKLEAK Mitigation of Such Bugs

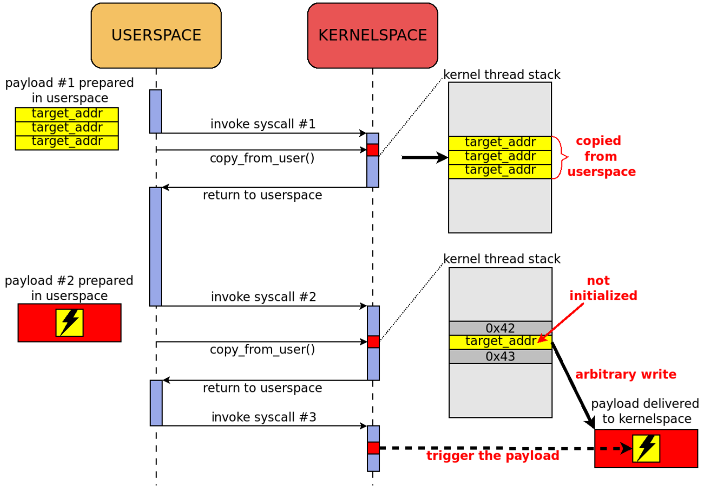


STACKLEAK Security Feature 2

- Blocks some* uninitialized kernel stack variable attacks
- Nice examples: CVE-2010-2963, CVE-2017-17712
- See cool write-up by Kees Cook:
  https://outflux.net/blog/archives/2010/10/19/cve-2010-2963-v4l-compat-exploit/

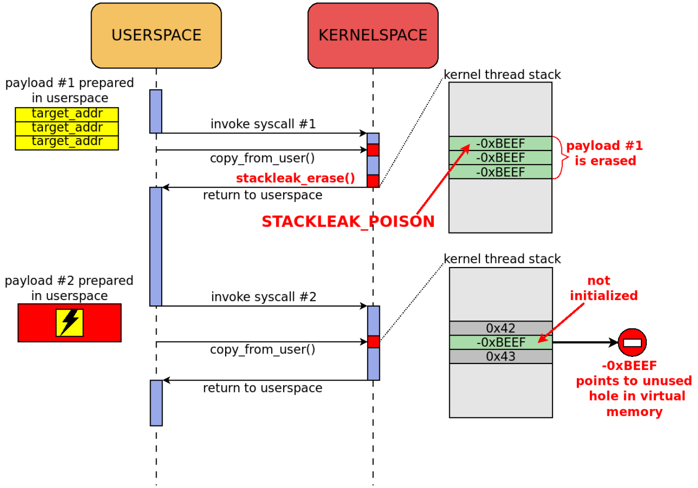
Uninitialized Stack Variable Attack

CVE-2010-2963 exploit



Mitigation of Uninitialized Stack Variable Attacks

CVE-2010-2963 exploit



Important Limitation

* STACKLEAK doesn't help against such attacks

during a **single** syscall

STACKLEAK Security Feature 3

Improves runtime detection of kernel stack depth overflow (blocks Stack Clash attack)

Interrelation of Security Mechanisms

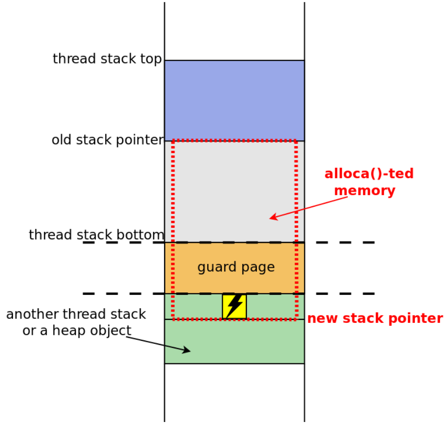
In mainline kernel STACKLEAK would be effective against kernel stack depth overflow only **in combination** with:

- `CONFIG_THREAD_INFO_IN_TASK`
- `CONFIG_VMAP_STACK` (kudos to Andy Lutomirski)

Viktor Vasnetsov, Bogatyrs (1898)

Stack Clash Attack for the Kernel Stack

Idea by Gael Delalleau: "Large memory management vulnerabilities" (2005)
Revisited in "The Stack Clash" by Qualys Research Team (2017)



STACKLEAK vs Stack Clash

- Read about STACKLEAK vs Stack Clash on grsecurity blog: https://grsecurity.net/an_ancient_kernel_hole_is_not_closed.php
- This code runs before each `alloca()` call:

```
	if (size >= stack_left) {
#if !defined(CONFIG_VMAP_STACK) && defined(CONFIG_SCHED_STACK_END_CHECK)
		panic("alloca() over the kernel stack boundary\n");
#else
		BUG();
#endif
	}
```

STACKLEAK vs Stack Clash

- Read about STACKLEAK vs Stack Clash on grsecurity blog: https://grsecurity.net/an_ancient_kernel_hole_is_not_closed.php
- This code runs before each `alloca()` call:

```
	if (size >= stack_left) {
#if !defined(CONFIG_VMAP_STACK) && defined(CONFIG_SCHED_STACK_END_CHECK)
		panic("alloca() over the kernel stack boundary\n");
#else
		BUG();
#endif
	}
```

- Hated by Linus

Cool, But What's the Price? (1)

Hardware: Intel Core i7-4770, 16 GB RAM
Performance test 1, attractive: building the Linux kernel

```
$ time make
```

```
Result on v4.18 defconfig:

    real 12m14.124s
    user 11m17.565s
    sys  1m6.943s

Result on v4.18 defconfig+stackleak:

    real 12m20.335s (+0.85%)
    user 11m23.283s
    sys  1m8.221s
```

Cool, But What's the Price? (2)

Hardware: Intel Core i7-4770, 16 GB RAM

Performance test 2, UNattractive:

```
$ hackbench -s 4096 -l 2000 -g 15 -f 25 -P
```

```
Average on v4.18 defconfig:  9.08s

Average on v4.18 defconfig+stackleak:  9.47s (+4.3%)
```

Cool, But What's the Price? (3)

Conclusion

STACKLEAK performance penalty varies for different workloads, so

1. Evaluate it on your expected workload before deploying in production (`STACKLEAK_METRICS` may help)
2. Decide whether it is fine in your case

Before Talking About the Upstreaming Process

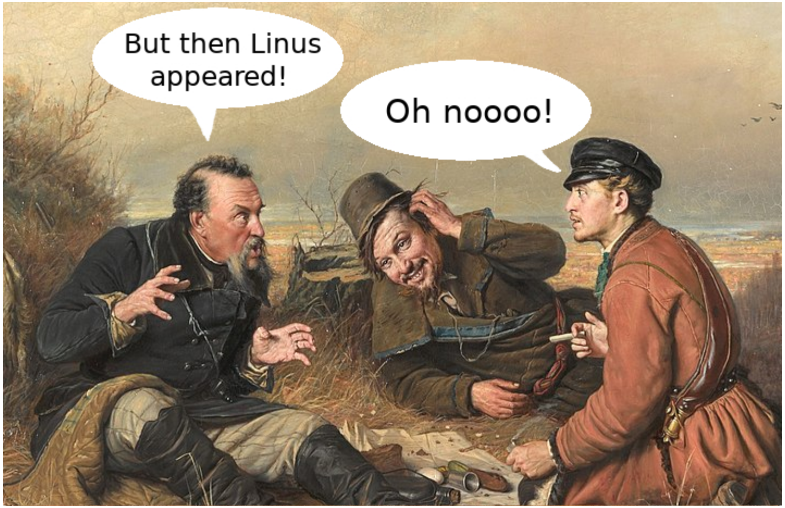
The STACKLEAK feature consists of:

- the code erasing the used part of the kernel thread stack
- the GCC plugin performing compile-time instrumentation for:
  - tracking the lowest border of the kernel stack
  - ~~`alloca()` check~~

STACKLEAK Upstreaming: a Long Thrilling Story

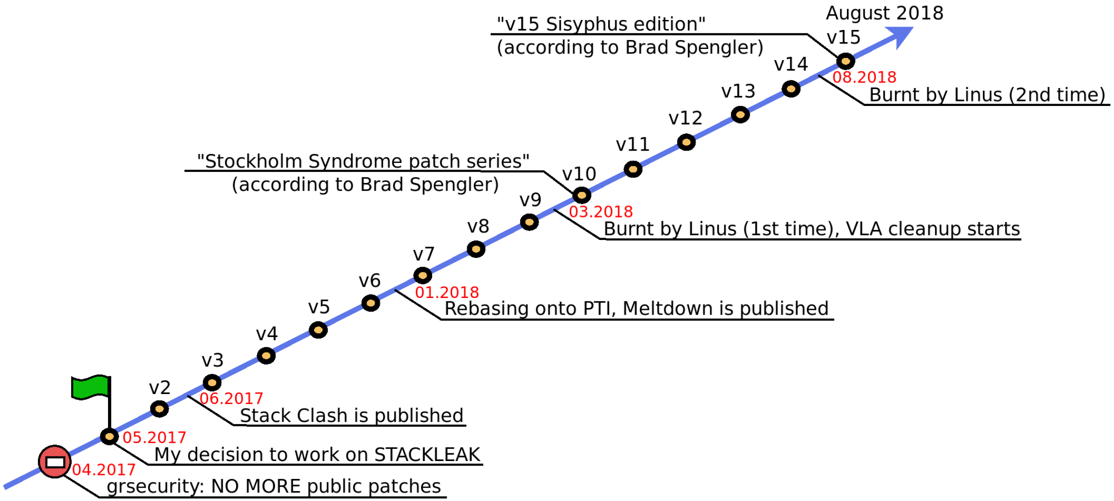
Vasily Perov, The Hunters at Rest (1871)

STACKLEAK Upstreaming: a Long Thrilling Story



Vasily Perov, The Hunters at Rest (1871)

STACKLEAK Upstreaming Timeline



August 2018
"v15 Sisyphus edition"
(according to Brad Spengler)
v15
08.2018
Burnt by Linus (2nd time)
v14
v13
v12
v11
"Stockholm Syndrome patch series"
(according to Brad Spengler)
v10
03.2018
Burnt by Linus (1st time), VLA cleanup starts
v9
v8
v7
01.2018
Rebasing onto PTI, Meltdown is published
v6
v5
v4
v3
06.2017
Stack Clash is published
v2
05.2017
My decision to work on STACKLEAK
04.2017
grsecurity: NO MORE public patches

STACKLEAK: Changes from the Original Version (1)

Bugs fixed in:

- original STACKLEAK gcc plugin
- original assertions in kernel stack tracking and `alloca()` check
- points of kernel stack erasing (found missing)

Plenty of refactoring:

- extracted the common part for easy porting to new platforms (includes rewriting of the stack erasing in C)
- got rid of hardcoded magic numbers, documented the code
- polished the codestyle until Ingo Molnar was satisfied (phew!)

STACKLEAK: Changes from the Original Version (2)

New functionality:

- x86_64 trampoline stack support
- tests for STACKLEAK (together with Tycho Andersen)
- arm64 support (by Laura Abbott)
- gcc-8 support in the plugin (together with Laura Abbott)

New functionality requested by Ingo Molnar:

- `CONFIG_STACKLEAK_METRICS` for performance evaluations
- `CONFIG_STACKLEAK_RUNTIME_DISABLE` (he forced me)

STACKLEAK: Changes from the Original Version (3)

Dropped functionality:

- assertions in stack tracking (erroneous)
- stack erasing point after ptrace/seccomp/auditing code at the beginning of syscall (hated by Linus)
- `alloca()` checking (hated by Linus):
  - `BUG_ON()` is now prohibited
  - all VLA (Variable Length Arrays) will be removed instead
  - and then global '-Wvla' flag will be set

https://patchwork.kernel.org/patch/10489873

STACKLEAK: Changes from the Original Version (4)

Brad Spengler

How security functionality will be properly implemented and maintained upstream if the maintainers don't understand what the code they've copy+pasted from grsecurity does in the first place

https://grsecurity.net/an_ancient_kernel_hole_is_not_closed.php

That is **not applicable** to STACKLEAK upstreaming efforts

What Does “Burnt by Linus” Mean?

- Strong language, even swearing (example)
- Technical objections are mixed with it
- NAKing without looking at the patches (example)
- Simply ignoring
- Maybe he is irritated with the kernel hardening initiatives **by default**?

https://en.wikipedia.org/wiki/File:Large_bonfire.jpg

What Does “Burnt by Linus” Mean?

- Strong language, even swearing (example)
- Technical objections are mixed with it
- NAKing without looking at the patches (example)
- Simply ignoring
- Maybe he is irritated with the kernel hardening initiatives **by default**?

https://en.wikipedia.org/wiki/File:Large_bonfire.jpg

I love the Linux kernel, but THAT kills my motivation

Sisyphus or Phoenix?

Will Linus finally merge STACKLEAK?

No?

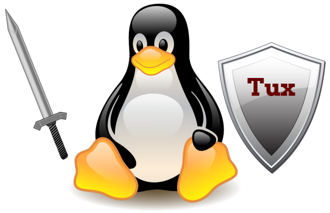
by Johann Vogel

Yes?

by Friedrich Justin Bertuch

Closing Thoughts

- WE are the Linux Kernel Community
- WE are responsible for servers, laptops, phones, PLCs, laser cutters, and other crazy things running GNU/Linux
- Let's put MORE effort into **Linux Kernel Security** – and **we will not be ignored!**



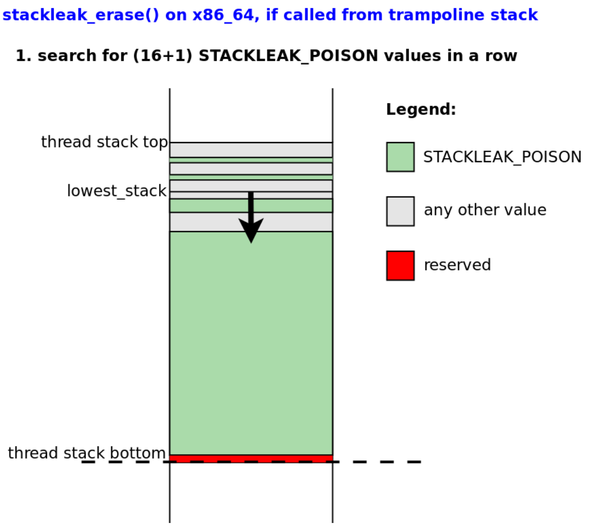
Thanks! Questions?

alex.popov@linux.com
@a13xp0p0v

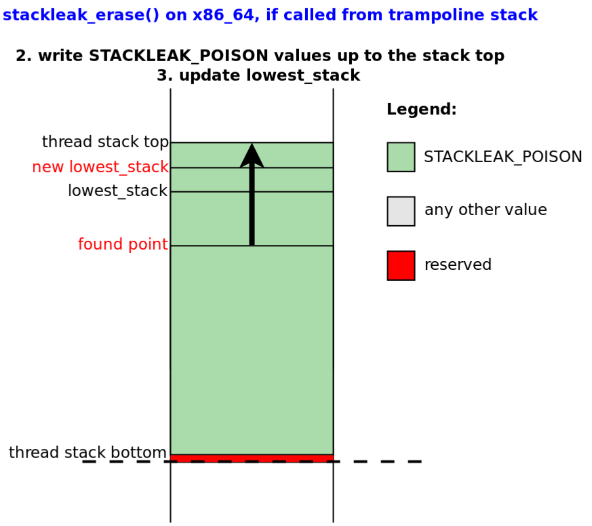
POSITIVE TECHNOLOGIES

http://blog.ptsecurity.com/
@ptsecurity

Erasing the Kernel Stack (1)


stackleak_erase() on x86_64, if called from trampoline stack
1. search for (16+1) STACKLEAK_POISON values in a row
thread stack top
lowest_stack
thread stack bottom
Legend:
STACKLEAK_POISON
any other value
reserved

Erasing the Kernel Stack (2)



Kernel Compile-Time Instrumentation

- Is done by STACKLEAK GCC plugin
- Inserts `stackleak_track_stack()` call for functions that:
  - have a **big stack frame**
  - call `alloca()` (have variable length arrays)
- ~~Inserts `stackleak_check_alloca()` call before `alloca()`~~