Between the Millstones: Lessons of Self-Funded Participation in Kernel Self Protection Project

Alexander Popov

Positive Technologies

October 22, 2018

About Me

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

Motivation of This Talk

Motivation

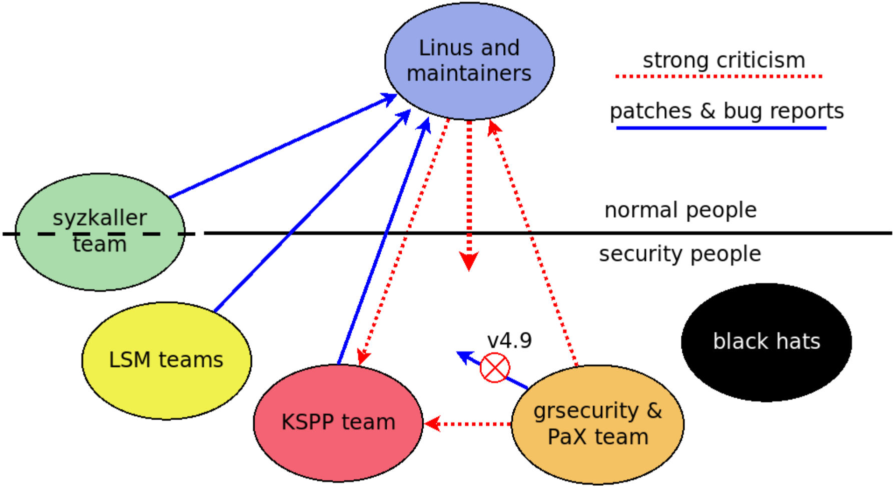
Today I see that the ideas from this talk could have been very useful for me 1.5 years ago, when I was beginning my participation in KSPP.

That's why I would like to share them.

Goals of This Talk

1. Involve more enthusiasts in Linux kernel security
2. Share the lessons I learned during kernel security development
3. Communicate on how we can improve our approaches

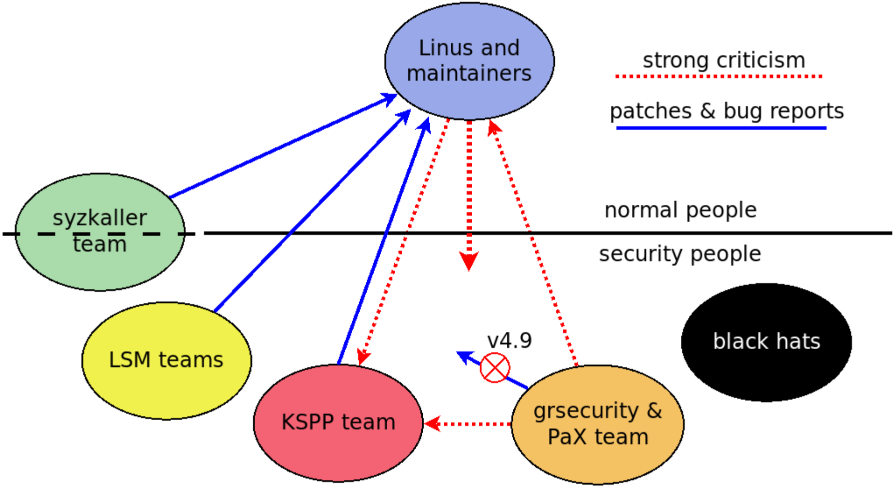
Who is Involved in Linux Kernel Security?



About LSM

- Linux Security Modules (LSM) is a framework that allows the Linux kernel to support a variety of computer security models
- LSM is primarily focused on supporting access control modules
- Projects: APPARMOR, SELINUX, SMACK, TOMOYO, YAMA...

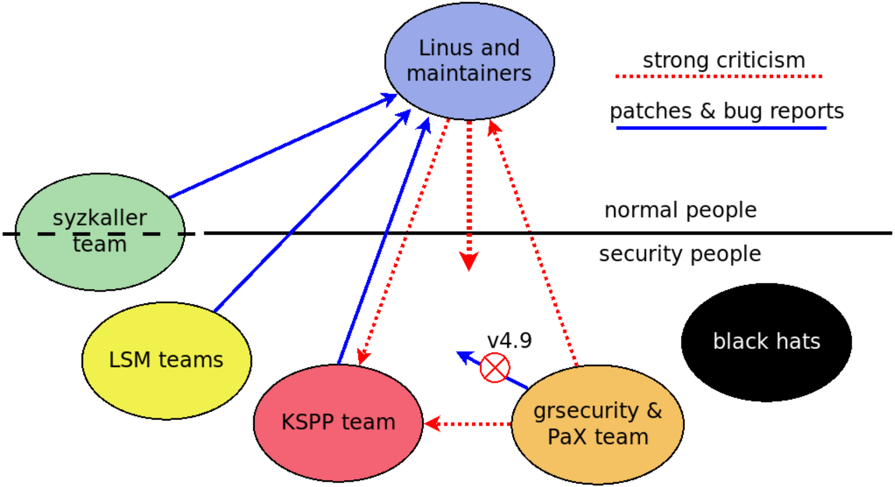
Who is Involved in Linux Kernel Security?



About syzkaller

- syzkaller is an unsupervised coverage-guided kernel fuzzer
- It gives great power in combination with sanitizers
- syzbot system uses syzkaller for continuous Linux kernel fuzzing
- It's an awesome project!
- Read the "Tale of thousand kernel bugs" by Dmitry Vyukov

Who is Involved in Linux Kernel Security?

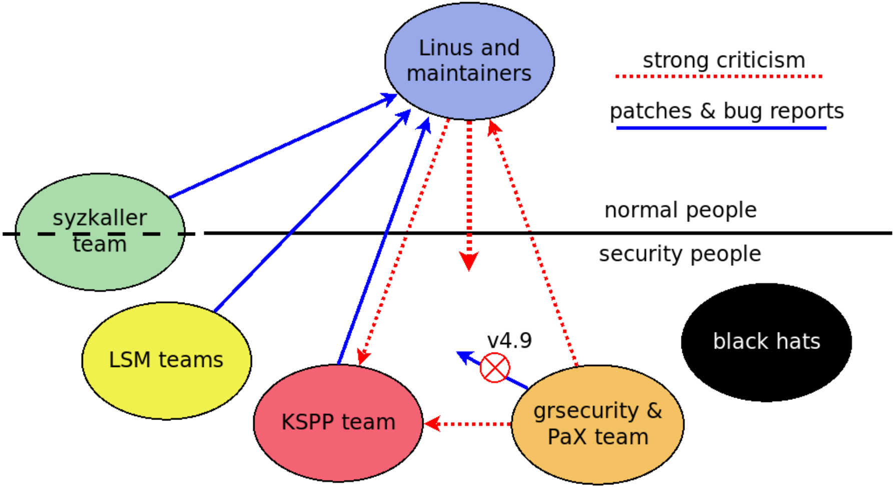


About grsecurity

- A patch for Linux kernel which provides security enhancements
- Includes PaX technologies
- Introduced a lot of excellent ideas to OS security world

  https://grsecurity.net/features.php
- But now is closed to the community (commercial secret)
- Last public version is for kernel 4.9 (April 2017)

Who is Involved in Linux Kernel Security?



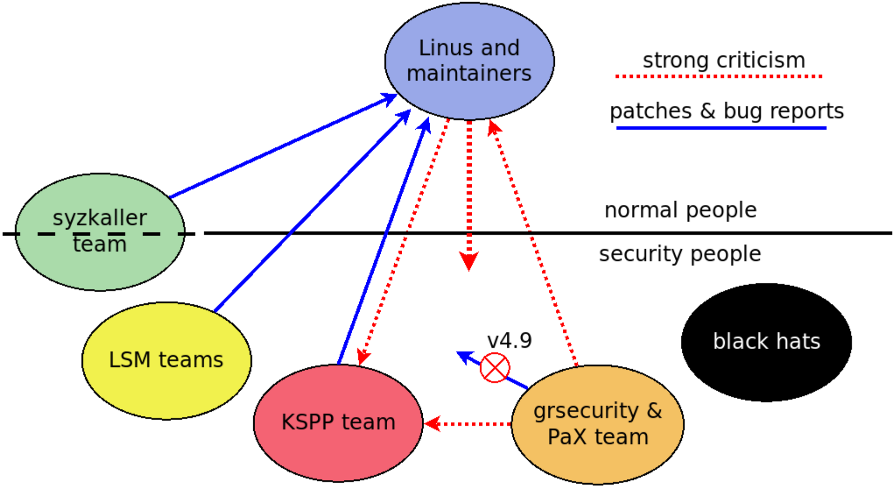
About Kernel Self Protection Project

- Security is more than fixing bugs
- Linux kernel should handle errors/attacks safely
- grsecurity & PaX ideas are the source of inspiration

KSPP goal

Eliminate vulnerability classes and exploitation methods in the Linux kernel **mainline**

Who is Involved in Linux Kernel Security?



Between the Millstones: That's How Mainline Hardening Is Made

https://foodal.com/kitchen/general-kitchenware/grain-mills/best-mills-reviewed/

Linux Kernel Self Protection

Linux kernel self protection is a very complex area, there are:

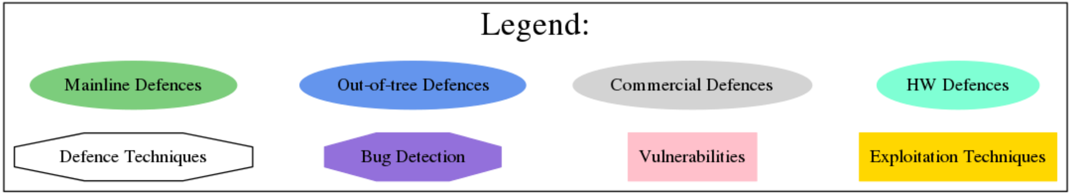
- Vulnerability classes
- Exploitation techniques
- Bug detection mechanisms
- Defence technologies
  - Mainline
  - Out-of-tree
  - Commercial
  - Provided by hardware

Drawn by Daniel Reeve, made by weta

And they all have complex relations...
It would be nice to have a **graphical representation** for easier navigating!

Linux Kernel Defence Map

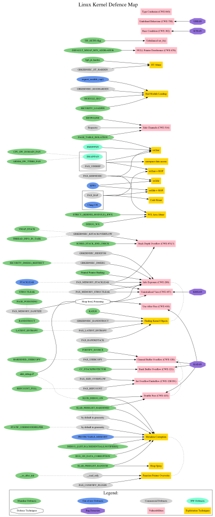
- So I created a Linux Kernel Defence Map
  https://github.com/a13xp0p0v/linux-kernel-defence-map
- Key concepts:



- Each connection between nodes represents a relationship
- N.B. This map doesn't cover cutting attack surface

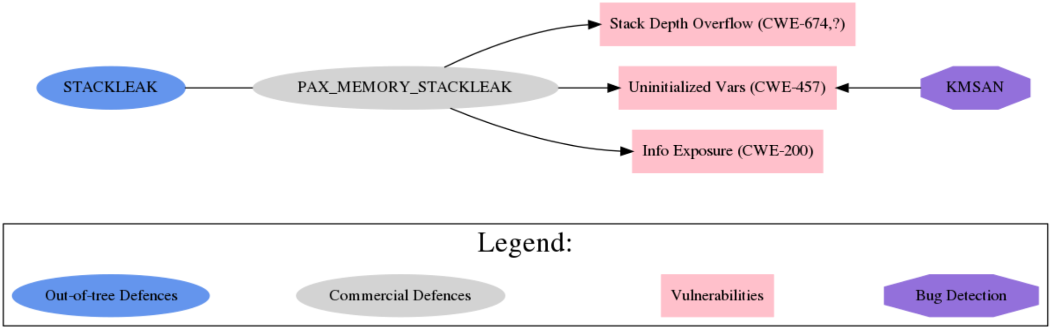
Linux Kernel Defence Map: Whole Picture

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



Linux Kernel Defence Map: STACKLEAK Part

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



Linux Kernel Defence Map: More Info

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

Got interested? Read the sources and start experimenting!

- grsecurity features
- Linux kernel security documentation
- Kernel Self Protection Project recommended settings
- Linux kernel mitigation checklist by Shawn C

Check the hardening options in your kernel `.config` with

https://github.com/a13xp0p0v/kconfig-hardened-check

My lessons from participation in KSPP

Story 1

Blocking consecutive double kfree()

CVE-2017-2636

- Once upon a time my customized syzkaller setup got a suspicious kernel oops
- I created a stable repro and found a race condition in `drivers/tty/n_hdlc.c`
- It caused a double-free bug, which I managed to exploit for LPE
- Debian, Ubuntu, Fedora, RHEL were affected (`CONFIG_N_HDLC=m`)

Responsible disclosure:
http://seclists.org/oss-sec/2017/q1/569

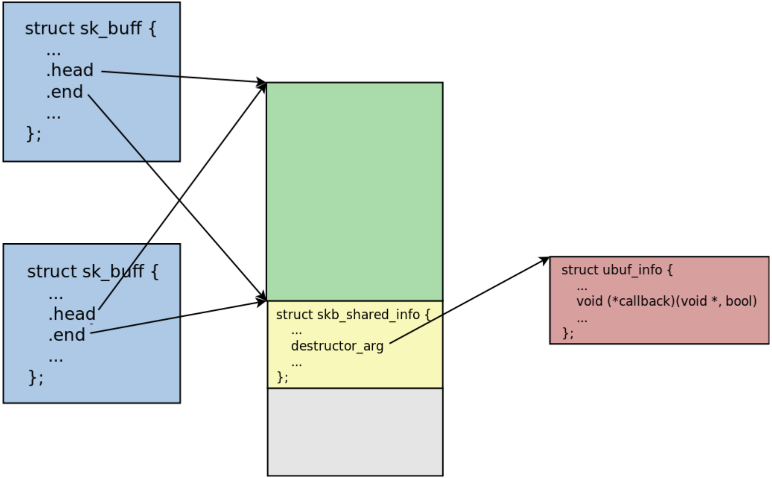
Detailed write-up about CVE-2017-2636 exploitation:
https://a13xp0p0v.github.io/2017/03/24/CVE-2017-2636.html

http://findwallpaper.info/street+racing+cars/page/7/

Surprise During PoC Development

- SLUB allocator accepts consecutive `kfree()` of the same address
- Kernel heap spraying after double-free gave me two `sk_buff`'s pointing to the same memory
- So double-free turns into use-after-free
- `slub_debug` detects this, but nobody uses it in production

Double-Free -> Use-After-Free on sk_buff



```
struct sk_buff {
    ...
    .head
    .end
    ...
};
```

```
struct sk_buff {
    ...
    .head
    .end
    ...
};
```

```
struct skb_shared_info {
    ...
    destructor_arg
    ...
};
```

```
struct ubuf_info {
    ...
    void (*callback)(void *, bool)
    ...
};
```

Blocking Consecutive Double-Free in SLUB

- I proposed a patch with a `BUG_ON()` similar to `fasttop` check in GNU C library allocator
- It provoked a lively discussion at LKML
- But finally this check got into the mainline kernel under `CONFIG_SLAB_FREELIST_HARDENED` (kudos to Kees Cook for his diplomacy)
- And today Ubuntu kernel has this option **enabled by default!**

Lessons From This Story

- Exploit practice can give interesting ideas for hardening
- Performance has the top priority for the Linux kernel maintainers
- But security can come under config options, distros enable them
- `BUG_ON()` provokes controversy [see the next slide]

About BUG_ON()

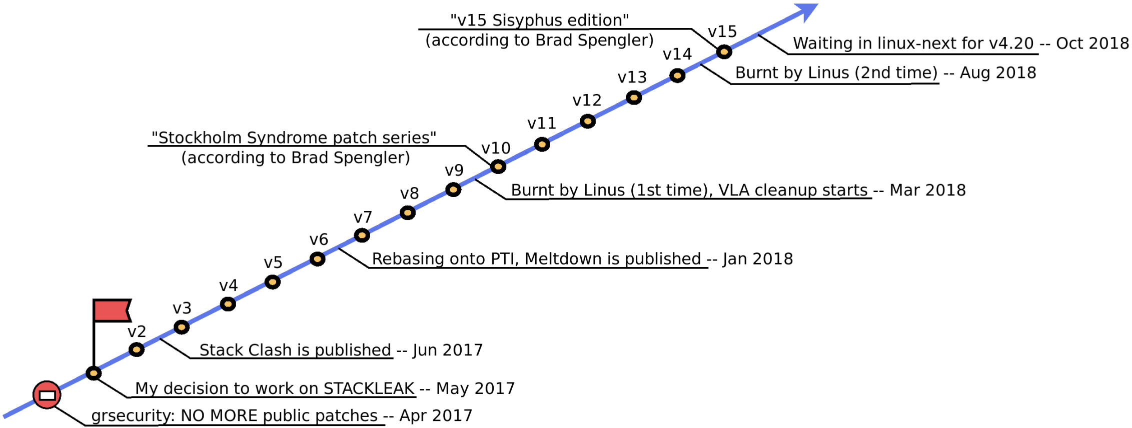
- Do your best to handle the error without `BUG_ON()`
- Think about using `WARN()`
- If you can't avoid `BUG_ON()`, **double-check** that you don't hold any core spinlocks, do see the oops and don't kill the whole machine. No, **triple-check**!
- Read these emails from Linus (several times):
  - "Just report it. Do no harm."
    https://lkml.org/lkml/2017/11/21/356
  - About `BUG_ON()` and locks
    http://lkml.iu.edu/hypermail/linux/kernel/1610.0/01217.html
  - `BUG_ON()` is forbidden for hardening (???)
    https://lkml.org/lkml/2018/8/15/450

My lessons from participation in KSPP

Story 2

Bringing PAX_MEMORY_STACKLEAK into the Linux kernel mainline

STACKLEAK Upstreaming



grsecurity: NO MORE public patches -- Apr 2017

My decision to work on STACKLEAK -- May 2017

Stack Clash is published -- Jun 2017

Rebasing onto PTI, Meltdown is published -- Jan 2018

Burnt by Linus (1st time), VLA cleanup starts -- Mar 2018

"Stockholm Syndrome patch series"
(according to Brad Spengler)

Burnt by Linus (2nd time) -- Aug 2018

"v15 Sisyphus edition"
(according to Brad Spengler)

Waiting in linux-next for v4.20 -- Oct 2018

STACKLEAK: Technical Details

- Recent patch series (v15):

  https://www.openwall.com/lists/kernel-hardening/2018/08/16/12
- Currently in `linux-next`, ready for the merge window
- Slides from the talk at LSS NA 2018:

  https://schd.ws/hosted_files/lssna18/b7/stackleak_LSS_NA_2018.pdf
- Article at LWN: https://lwn.net/Articles/764325/
- Dispute with Brad Spengler: https://lwn.net/Articles/764685/

STACKLEAK Lessons: What Works Well

1. Cover letter describing the goal, benefits, performance impact
2. Release early, release often (RERO)
    - RFC tag for early versions of the patch series
    - TODO list and changelog in the cover letter
3. Careful handling of the feedback from the community and Brad
4. Cool-headed separating technical arguments from personal attacks
5. Flexibility **and** persistence

KSPP Motto

From Terminator 2: Judgment Day

STACKLEAK Lessons: What Doesn't Work

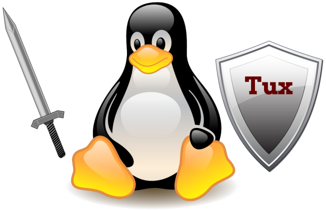
1. Illusions that my work will be appreciated
2. Not expanding the list of recipients as development progresses
3. It looks like KSPP roadmap is not coordinated with Linus
   - The risk of getting NAK after a year of hard work
   - The lack of clear rules for hardening patches, e.g. about:
     - Assembly language usage
     - Runtime disabling of the feature
     - `BUG_ON()` usage
4. Not knowing Monty Python comedy ;)
   https://lkml.org/lkml/2018/8/15/510

How Can We Do Better?

- Working harder, of course!
- [?] Having a list of kernel hardening “behavior patterns” approved by maintainers
- [?] Having the KSPP roadmap coordinated with maintainers
- [?] Large companies/organizations explicitly requesting/promoting concrete kernel hardening features
- More enthusiastic people participating, for sure!

Closing Thoughts

- Linux kernel development is very interesting
- Linux kernel hacking and hardening is TWICE as interesting and sometimes dangerous :)
- But HERE you can find BIG challenges and get joy in the battle!



Thanks! Questions?

alex.popov@linux.com
@a13xp0p0v

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

POSITIVE TECHNOLOGIES