Maintaining the Unmaintainable: Picking Up the Baton of a Secure Kernel Patchset

Matthew Ruffell | matthew@ruffell.nz | https://ruffell.nz

LINUX.CONF.AU
21-25 January
2019
Christchurch, NZ

Declaration

- This talk reflects the status and features of dapper-secure-kernel-patchset(-stable)
- This talk DOES NOT represent the current status and features of the grsecurity patchset, by Open Source Security Inc
- There will be overlap, this is due to dapper-secure-kernel-patchset(-stable) being a fork which branched two years ago
- I am not affiliated with Open Source Security Inc, or have anything to do with the company or current grsecurity patchset

Introduction

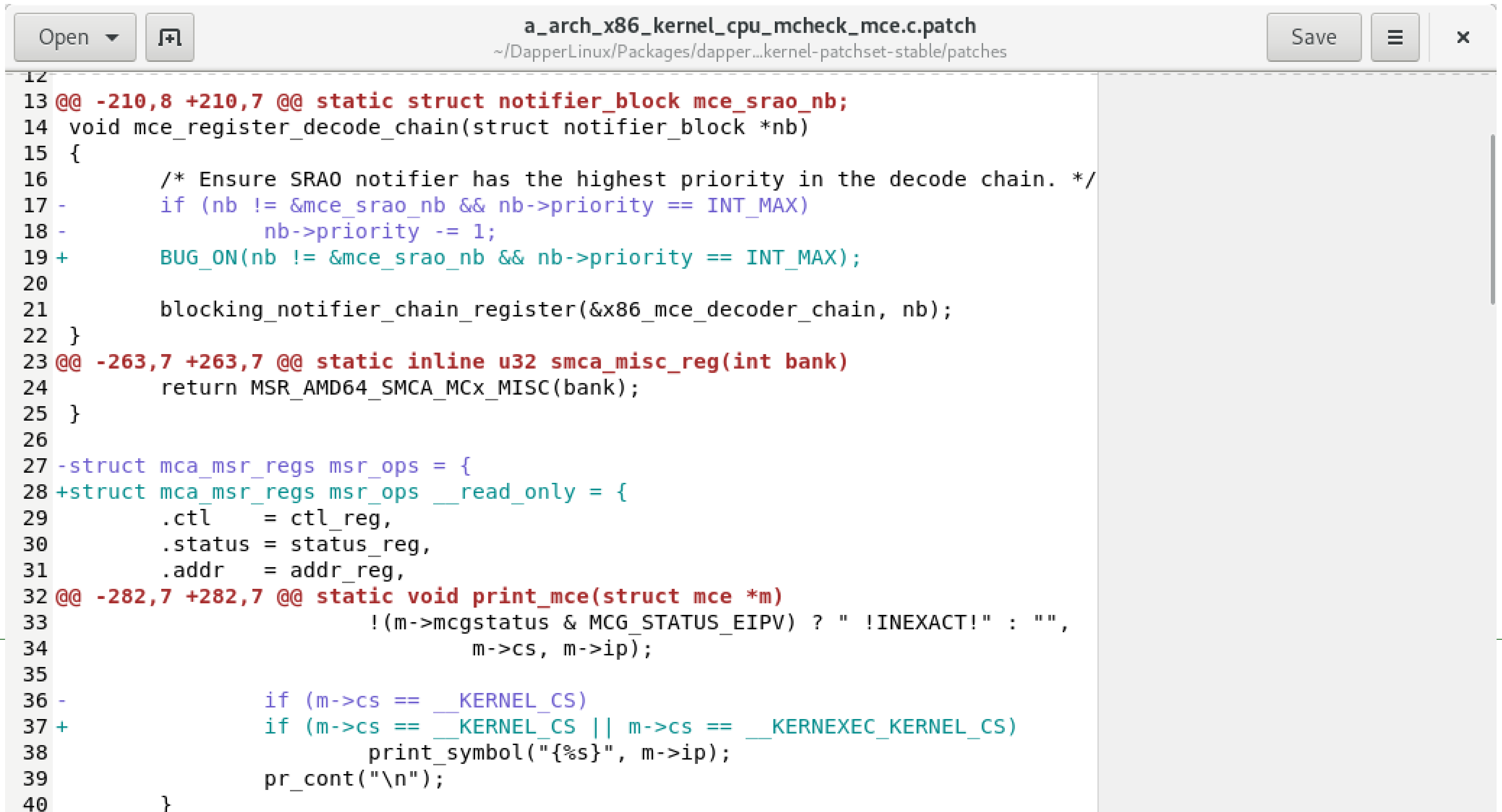
- There is a kernel patchset called grsecurity, which focuses on improving security
- Made by Open Source Security Inc
- Been around for a long time, contained many technological breakthroughs, like ASLR, SMEP, ...
- Open Source Security Inc decided to remove public patches, suggest community maintain it themselves
- I was developing Dapper Linux at the time, and depended on patchset. Big spanner in the works, and something had to be done about it.
- I learned kernel dev and attempted to maintain it for a few years
- I ended up being the last maintainer for a public release

What Is The Purpose Of This Patchset?

- To go above and beyond upstream for security and hardening
- Focuses not so much on fixing individual instances of bugs, but eliminating entire classes of vulnerabilities
- Attaining self defence against unknown exploits
- Because of this, a patched kernel is *typically* not affected by the latest exploit
- Idea is to increase the cost of exploitation, as normal exploits fail, so exploits which work are kept secret and risk / reward is higher to reveal and use a working exploit for attack

Why Is This Patchset Scary?

- Comes as a single monolithic patch.
- 10.3mb in size, 253,675 LOC, 113,845 insertions, 32,669 deletions
- Spans over 3422 files



Open | a_arch_x86_kernel_cpu_mcheck_mce.c.patch | ~/DapperLinux/Packages/dapper...kernel-patchset-stable/patches | Save

```
@@ -210,8 +210,7 @@ static struct notifier_block mce_srao_nb;
 void mce_register_decode_chain(struct notifier_block *nb)
 {
        /* Ensure SRAO notifier has the highest priority in the decode chain. */
-       if (nb != &mce_srao_nb && nb->priority == INT_MAX)
-               nb->priority -= 1;
+       BUG_ON(nb != &mce_srao_nb && nb->priority == INT_MAX);
 
        blocking_notifier_chain_register(&x86_mce_decoder_chain, nb);
 }
@@ -263,7 +263,7 @@ static inline u32 smca_misc_reg(int bank)
        return MSR_AMD64_SMCA_MCx_MISC(bank);
 }
 
-struct mca_msr_regs msr_ops = {
+struct mca_msr_regs msr_ops __read_only = {
        .ctl    = ctl_reg,
        .status = status_reg,
        .addr   = addr_reg,
@@ -282,7 +282,7 @@ static void print_mce(struct mce *m)
                        !(m->mcgstatus & MCG_STATUS_EIPV) ? " !INEXACT!" : "",
                                m->cs, m->ip);
 
-               if (m->cs == __KERNEL_CS)
+               if (m->cs == __KERNEL_CS || m->cs == __KERNEXEC_KERNEL_CS)
                        print_symbol("{%s}", m->ip);
                pr_cont("\n");
        }
```

Why Is This Patchset Scary?

- Documentation is sparse, sometimes out of date
  - Text files by The PaX Team explaining features in detail
  - Commit messages in Minipli's fork
  - Kconfig entries
  - Whatever you can find in mailing lists / forums
- When compiled, tends to break userspace a lot, inexperienced users can possibly break their install
- Patchset focuses on technological greatness, ignores maintainability completely
  - Not a lot of comments

Feature Overview

- So you are probably wondering how most classes of vulnerabilities are eliminated
- Patchset has three major categories of features:
  - GCC Plugins
  - Core kernel enhancements
  - Misc fixes
- Many break userspace or have significant performance impact

GCC Plugins

- GCC plugins are intended to greatly improve security for the least amount of effort
- Be easy to maintain, and are portable (different $ARCH, programs other than kernel, e.g. Coreutils)
- Reduce the amount of work through automation
- Keep the patchset smaller since trivial patches are not needed

PAX_CONSTIFY_PLUGIN

- For structs that ONLY contain function pointers (ops)
- Automatically make all fields constant
- Ensures function pointers cannot be modified or overwritten
- __no_const && __do_const annotations

GRKERNSEC_RANDSTRUCT

- For structs that ONLY contain function pointers (ops)
- Randomises layouts of structs
- Can add requirement of needing an infoleak to derandomise function locations
- Not as effective on distro kernels as seed is known, good for self compile
  - Distros: each build still has different seed, still has some use
  - Exploit must be tailored to each build
- __randomize_layout annotations
- Upstreamed by Kees Cook in 4.13

PAX_MEMORY_STACKLEAK

- Right before a syscall returns, kernel stack used in syscall is erased
- Prevents infoleaks from uninitialised variables left on kernel stack and removes secrets faster
- Upstreamed by Alexander Popov in 4.20

PAX_MEMORY_STRUCTLEAK

- When a struct is being copied from kernel space to user space, uninitialised variables will be initialised to zero
- Prevents information leaks of kernel addresses to userland
- Upstreamed by Kees Cook in 4.11

PAX_SIZE_OVERFLOW

- Detects and reports integer overflow / underflow so they can be fixed
- Instruments code with double wide integer type depending on arch and variable
- Overflows and underflows can be found by examining higher bits
- Logs to dmesg and sends SIGKILL to process to prevent exploitation
- __intentional_overflow annotations, used for timers / counters intended to wrap around
- Cons: Requires hash table of all functions in kernel, must be generated each release

PAX_RAP

- Prevents code reuse during exploitation (ROP, JOP)
- Two major features:
  - Implements forward and backward edge CFI (Control Flow Integrity)
    - Basically calculates a hash of functions we are allowed to jump to and return back to
    - If we try to return to a different function with a different hash, then execution stops
  - Implements a probabilistic guarantee that we return back to intended target
    - "Encrypts" (XOR) the return address and places on stack
    - Upon return, address is “decrypted” (XOR)
    - Compared with actual return address. If they don't match, execution stops
    - Probabilistic since key vulnerable to leaking from its reserved register

Kernel Enhancements

- Usually fully fledged features which are implemented into the kernel
- Mosty #ifdef && #endif with CONFIG_$FEATURE flags
- Tightly coupled to the subsystem or feature being hardened

PAX_MPROTECT

- Hardens the MPROTECT syscall
- MPROTECT will no longer:
  - make pages executable when they were not executable upon creation
  - make read only executable pages writeable
  - create executable pages from anonymous memory
  - making read only after relocations data pages writeable again
- The feature that is most seen by grsec users, as it breaks userspace!
- Python, Java, Gnome-Shell, Firefox, all need this feature disabled via file attrs

PAX_KERNEXEC

- Enforces W^X kernel pages, similar to PAX_MPROTECT hardening
- Extends W^X to loadable modules

PAX_MEMORY_UDEREF

- Prevents the kernel from dereferencing user space pointers when kernel expects kernel space pointers
- Stops kernel execution flow from going into user space
- Prevents all ret2usr / ret2dir exploits
- x86_64 has per-cpu-pgd (page global directory) (KPTI dabbles in this, more later)
- Dedicated pgd for kernel space and user space (KPTI implements this)

PAX_ASLR

- Randomises base address of processes on each execution
- Prevents exploitation where addresses must be known (stack overflows)
- Has been extended to KASLR upstream, randomises kernel location
- PAX_RANDKSTACK
  - Randomises the kernel stack base
- PAX_RANDUSTACK
  - Randomises the stack on userspace applications

PAX_MEMORY_SANITIZE

- Kernel will erase all memory pages and slab objects as soon as they are freed
- Reduces lifetime of secrets
- Detects use after free on structs containing pointers, as a deref on erased memory causes access violation

PAX_REFCOUNT

- Detects and prevents reference counters from overflowing
- Overflowing reference counters are sometimes freed when overflown, while still in use, leading to exploitable conditions
- atomic_unchecked_t, atomic_unchecked_add, atomic_unchecked_dec, ...
- Upstreamed by many awesome developers in Linux 4.11, 4.13 and 4.15
- refcount_t in upstream

PAX_USERCOPY

- Make kernel enforce fixed sizes when copying objects between kernel space and user space in both directions
- Prevents information leaks from uninitialized data when too much kernel space data is requested by userspace
- Prevents kernel heap overflows

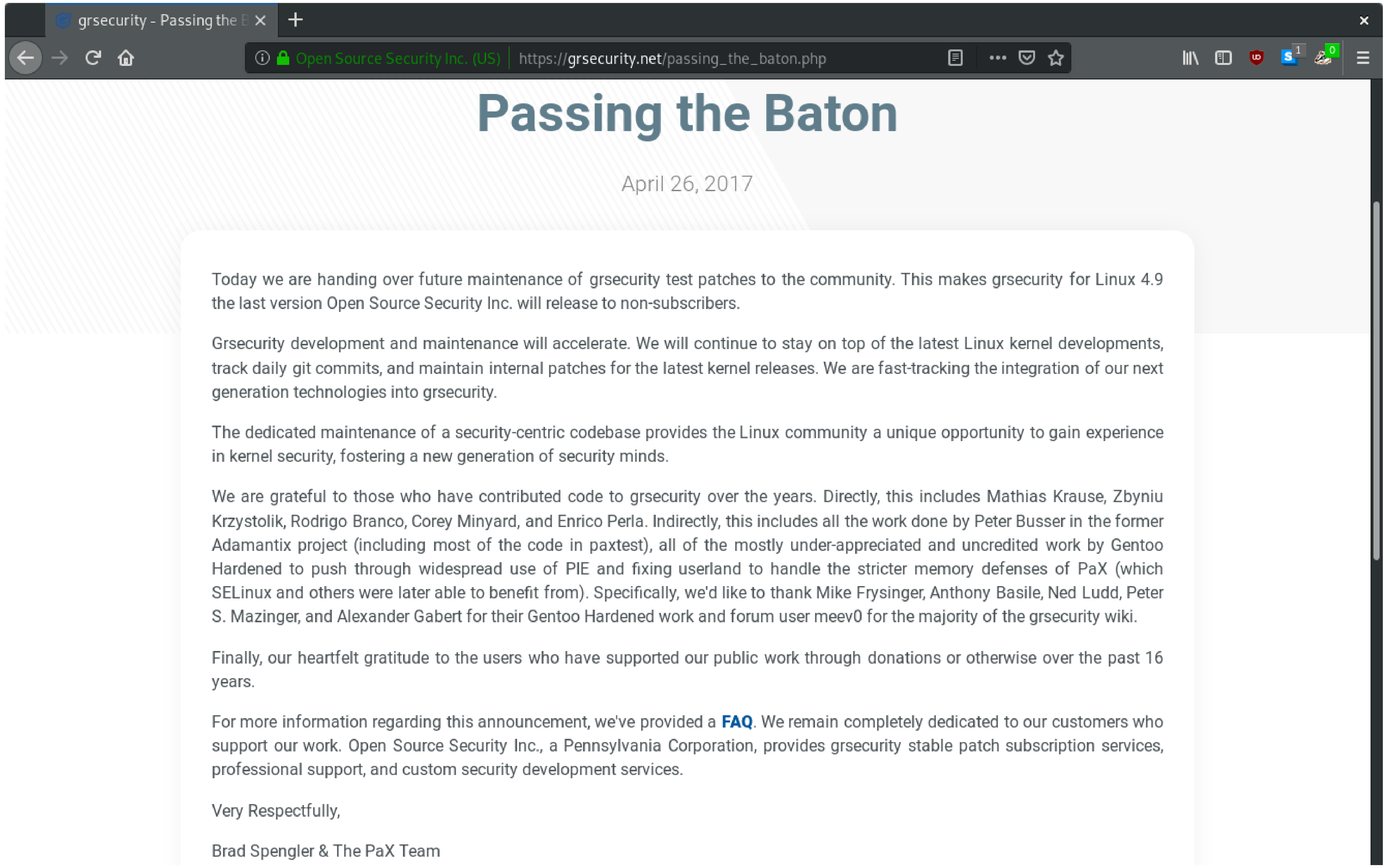
GRKERNSEC_CHROOT

- Locks down and enforces strict access controls on chroot jails
- Cons: Can't turn off for specific chroots, as is compiled into kernel.
- Must have it enabled for all chroots or for none at all
- GRKERNSEC_CHROOT_MOUNT
  - Disable mounting from within chroot
- GRKERNSEC_CHROOT_DOUBLE
  - Prevent chroot inside of chroot, used for escaping
- GRKERNSEC_CHROOT_CHDIR
  - Change working dir of new chroot to root of chroot. Prevents escape with ..
- GRKERNSEC_CHROOT_CHMOD
  - Prevent chmod attributes, stops suid / guid being set
- GRKERNSEC_CHROOT_SHMAT
  - Prevent chroot from accessing shared memory segments
- GRKERNSEC_CHROOT_UNIX
  - Prevent chroot from connecting to unix sockets bound outside chroot
- GRKERNSEC_CHROOT_FINDTASK
  - Prevent kill, signals, ptrace, or viewing any process outside chroot
- GRKERNSEC_CHROOT_NICE
  - Prevent changing priority of process outside chroot

Misc Fixes

- Bugfixes for problems noticed during forward porting / watching daily commits
- Sometimes small type fixes, logic fixes or, sometimes serious zero days
- Typically small in size
- Many of them have now been upstreamed

Announcement From Open Source Security Inc


grsecurity - Passing the B
Open Source Security Inc. (US)
https://grsecurity.net/passing_the_baton.php

Passing the Baton

April 26, 2017

Today we are handing over future maintenance of grsecurity test patches to the community. This makes grsecurity for Linux 4.9 the last version Open Source Security Inc. will release to non-subscribers.

Grsecurity development and maintenance will accelerate. We will continue to stay on top of the latest Linux kernel developments, track daily git commits, and maintain internal patches for the latest kernel releases. We are fast-tracking the integration of our next generation technologies into grsecurity.

The dedicated maintenance of a security-centric codebase provides the Linux community a unique opportunity to gain experience in kernel security, fostering a new generation of security minds.

We are grateful to those who have contributed code to grsecurity over the years. Directly, this includes Mathias Krause, Zbyniu Krzystolik, Rodrigo Branco, Corey Minyard, and Enrico Perla. Indirectly, this includes all the work done by Peter Busser in the former Adamantix project (including most of the code in paxtest), all of the mostly under-appreciated and uncredited work by Gentoo Hardened to push through widespread use of PIE and fixing userland to handle the stricter memory defenses of PaX (which SELinux and others were later able to benefit from). Specifically, we'd like to thank Mike Frysinger, Anthony Basile, Ned Ludd, Peter S. Mazinger, and Alexander Gabert for their Gentoo Hardened work and forum user meev0 for the majority of the grsecurity wiki.

Finally, our heartfelt gratitude to the users who have supported our public work through donations or otherwise over the past 16 years.

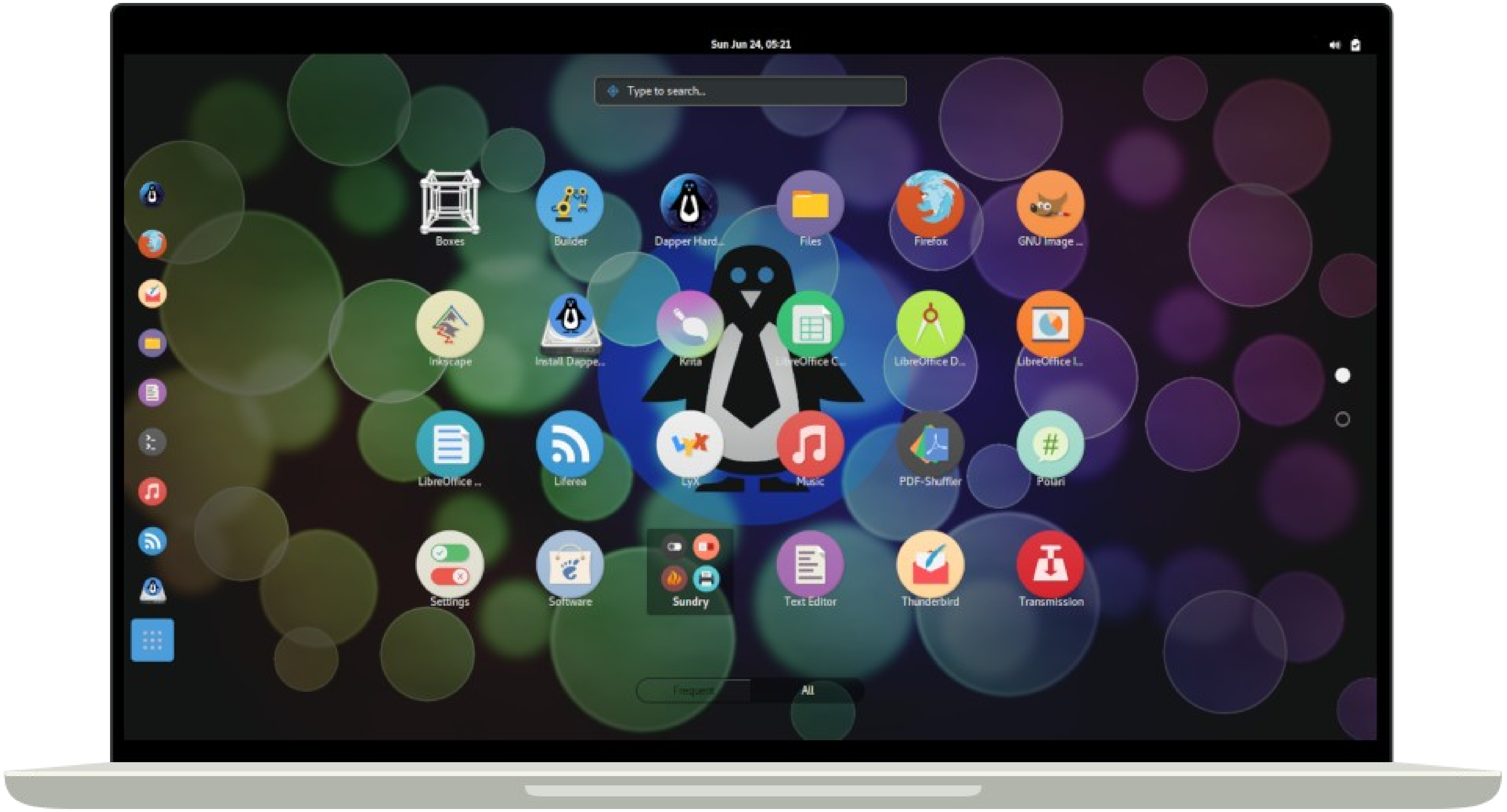
For more information regarding this announcement, we've provided a **FAQ**. We remain completely dedicated to our customers who support our work. Open Source Security Inc., a Pennsylvania Corporation, provides grsecurity stable patch subscription services, professional support, and custom security development services.

Very Respectfully,

Brad Spengler & The PaX Team

Announcement From Open Source Security Inc

- "Today we are handing over future maintenance of grsecurity test patches to the community."
- "The dedicated maintenance of a security-centric codebase provides the Linux community a unique opportunity to gain experience in kernel security, fostering a new generation of security minds."
- Left only previous patch to 4.9.24
- Pretty clear there is some large shoes to fill.



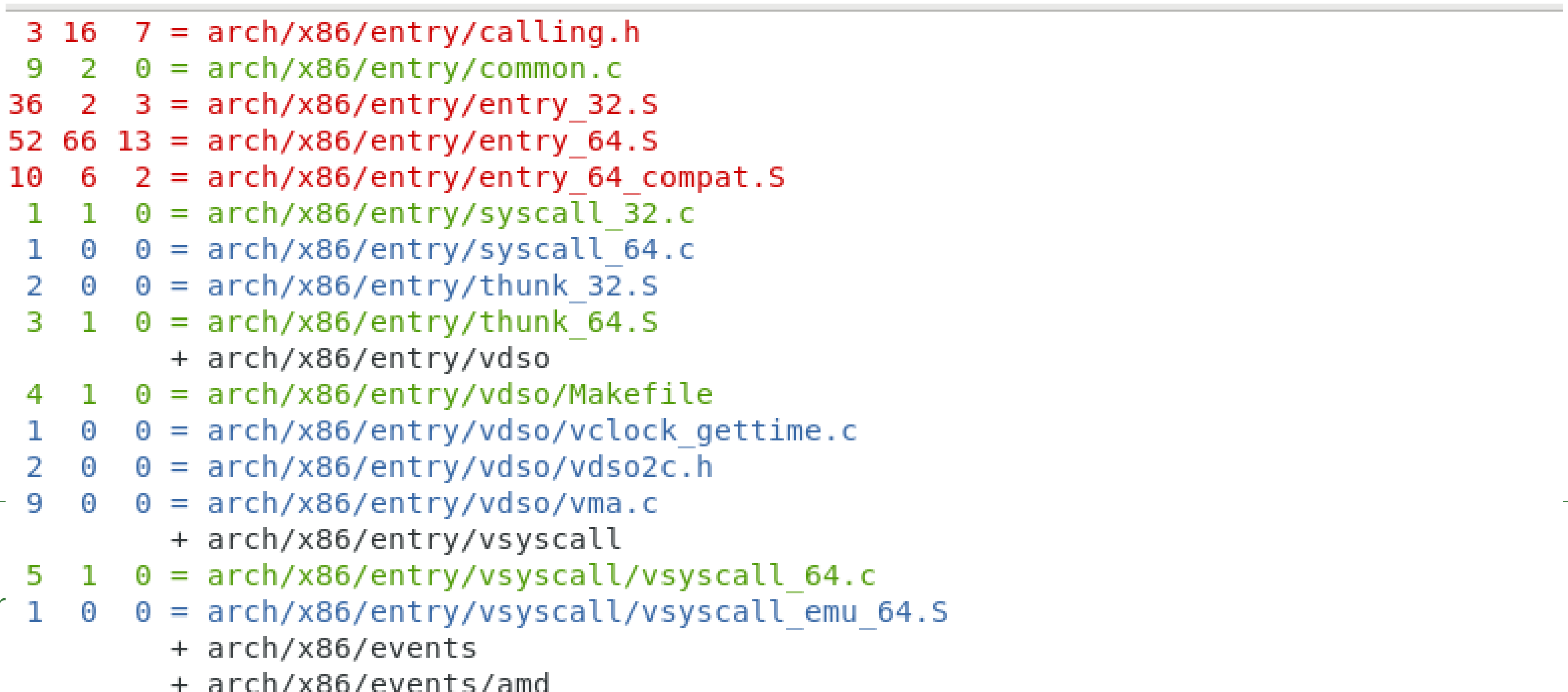
Dapper Linux

Feeling Left High And Dry

- Feeling left high and dry, and unwilling to give up on the technological advancements that the patchset provides
- I thought the patchset was pretty cool, and wanted to learn more about it
- I decided to learn kernel development, and attempt to maintain the patchset as a complete kernel newbie.
- Treated it as a call to adventure

Attempt To Forward Port To Major Kernel Version

- I split the monolithic patch into individual files
- Started naive attempts to automate, found wiggle
- When fixing compile errors, found incorrect fuzzy placement, forced to abandon
- Set patch fuzz to 0, look at conflicts between major versions, fix manually



```
 3 16  7 = arch/x86/entry/calling.h
 9  2  0 = arch/x86/entry/common.c
36  2  3 = arch/x86/entry/entry_32.S
52 66 13 = arch/x86/entry/entry_64.S
10  6  2 = arch/x86/entry/entry_64_compat.S
 1  1  0 = arch/x86/entry/syscall_32.c
 1  0  0 = arch/x86/entry/syscall_64.c
 2  0  0 = arch/x86/entry/thunk_32.S
 3  1  0 = arch/x86/entry/thunk_64.S
         + arch/x86/entry/vdso
 4  1  0 = arch/x86/entry/vdso/Makefile
 1  0  0 = arch/x86/entry/vdso/vclock_gettime.c
 2  0  0 = arch/x86/entry/vdso/vdso2c.h
 9  0  0 = arch/x86/entry/vdso/vma.c
         + arch/x86/entry/vsyscall
 5  1  0 = arch/x86/entry/vsyscall/vsyscall_64.c
 1  0  0 = arch/x86/entry/vsyscall/vsyscall_emu_64.S
         + arch/x86/events
         + arch/x86/events/amd
```

Numbers On Conflicts Between Versions

- 4.9 -> 4.10, Status: Skipped. 4.11 already out.
- 4.9 -> 4.11, Status: Complete, compiles, boot fails.
  - Conflicts: 75 files, 134 hunks. 221 files changed, 1258 insertions, 6808 deletions
- 4.11 -> 4.12, Status: Incomplete, left most for 4.13.
  - Conflicts: 257 files, 560 hunks. 70 files changed, 340 insertions, 796 deletions
- 4.12 -> 4.13, Status Complete, does not compile.
  - Conflicts: 461 files, 1337 hunks. 755 files changed, 3390 insertions, 27320 deletions
- 4.13 -> 4.14, Status: Incomplete.
  - Conflicts: 296 files, 493 hunks. 215 files changed, 846 insertions, 1662 deletions

Forced To Rethink Strategy

- Forward porting major versions is an extremely hard task
- Some parts require extensive rewrites and reworking, upstream changes and gets re-factored continuously
- As time passes, maintenance effort increases
- Some parts trivial and extremely boring, but cant trust automated tools due to fuzz mistakes
- I couldn't handle getting one major release even working, and I attempted to forward port 5 major versions
- A new major Linux was released before I managed to forward port to the last one.
- Spender and The PaX Team have done every version from Linux 2.4, an extremely impressive effort

Major Versions Will Not Work, Maintain LTS Instead

- LTS minor releases are much smaller and manageable
- New point release:
  - Read Greg KH thread on announce mailing list
  - Get a diff, read all diffs
  - Patch patchset ontop of new point release, see conflicts
  - Repair conflicts
  - Generate test patch, check conflicts and compile errors
  - Generate release patch
  - Sign and release
  - Compile kernel
  - Distribute out packages (RPMs for Fedora based systems, a user also packaged debs, and another covered gentoo)
- This happens twice a week, although I sometimes waited until the weekend
- Just maintaining LTS point releases is a lot of work!

Everything Was Fine Until...

- Point releases started to get larger, and larger ...
- GCC8 had some complications with gcc-plugins
- Meltdown and Spectre really, really threw a spanner in the works
- UDEREF has a per-cpu-pgd, and KPTI also starts to implement this, UDEREF incompatible with Meltdown / Spectre mitigations, and requires a lot of rework
- Had to change how I maintained the kernel, had to cherry pick patches and revert all Meltdown and Spectre mitigations
- Users had to decide between kernel hardening or Meltdown and Spectre mitigations - most chose to move upstream and stop using my patchset

Timeline Of Maintainers

- Hardened Gentoo dropped patches 19 August 2017 - used grsec patch
- Alpine final release 27 Nov 2017 - Linux 4.9.65 - used their own forward port, then changed to minipli
- Minipli final release 4 Jan 2018 - Linux 4.9.74, 50 releases
- Myself, Matthew Ruffell final release 26 October 2018 - Linux 4.9.135, 111 releases

Future Of The Patchset

- If I really wanted to, I could continue maintenance, but it is a huge time sink for very few users
- In reality, the patchset is experiencing advanced bitrot
- There are two scenarios
  - Split up all features into separate files, continue to maintain as one large set
  - Cherry pick specific features, upstream them

What Should Be Kept In The Future

- All the GCC plugins are prime candidates for maintaining and upstreaming, have the most benefit for least work
- I wont promise upstreaming anything, but I am interested in SIZE_OVERFLOW, PAX_MEMORY_SANITIZE, PAX_MPROTECT

Lessons Learned

- Wise words from Kees Cook: "Forks are always a risk", "there will always be another wall"
- You have it within you to maintain your favourite projects for a short while
- Upstreaming will always have a larger impact and lifespan than forks
- Maintaining this patchset has taught me so much about most subsystems and how the kernel works

Greetz

- Spender and The PaX Team
  - Thanks for your hard work over the years, and for keeping releases free as long as you did
- Kees Cook
  - Thank you for all your upstreaming efforts, I have read many, many of your commits when hunting down merge conflicts
- Alexander Popov
  - Congratulations for upstreaming StackLeak

Where To Find The Patchsets

- https://dapperlinux.com/patchset.html
- https://github.com/dapperlinux/dapper-secure-kernel-patchset
- https://github.com/dapperlinux/dapper-secure-kernel-patchset-stable

Matthew Ruffell

matthew@ruffell.nz

https://ruffell.nz