The Kernel Self-Protection Project and how you can help

Gustavo A. R. Silva
gustavoars@kernel.org
@embeddedgus

Supported by The Linux Foundation & Google

Kernel Recipes
June 2, 2022
Paris, France

Who am I?

- Embedded Systems.
- RTOS & Embedded Linux.
- Upstream first – 6 years.
- Kernel developer & maintainer.
- GOSST - Linux kernel division.
- Volunteer at @kidsoncomputers

Who am I?

- Embedded Systems.
- RTOS & Embedded Linux.
- Upstream first – 6 years.
- Kernel developer & maintainer.
- GOSST - Linux kernel division.
- Volunteer at @kidsoncomputers

“Tuxote”

Agenda

- The Kernel Self-Protection Project
- Work in progress and some accomplishments.
- How you can help. :)
- Conclusions.

The Kernel Self-Protection Project

The Kernel Self-Protection Project

- It’s an **upstream project**. Not a random downstream clone of Linux.
- Focused on **hardening** the upstream Linux kernel.
- We want to eliminate entire **bug classes** and **methods of exploitation**.
- Developing of defense mechanisms that **cut off whole classes of vulnerabilities**. Best way to approach the problem of security.
- Moving the codebase to use **safer APIs**.
- **Not** about writing CVEs.

Tools and Platforms

linux-hardening

- **Upstream** mailing list. Created in 2020.
- Needed a list to discuss development, maintenance and **all things related**.
- Old list (kernel-hardening) only wanted new stuff.
- A place to discuss about all the small **details and middle steps** in the process of hardening the kernel was needed.
- linux-hardening@vger.kernel.org
- https://lore.kernel.org/linux-hardening/202009281907.946FBE7B@keescook/

Patchwork

- Keep track of tags: Reviewed-by, Tested-by, Acked-by, etc.
- The KSPP is not a subsystem, but it has **maintainers**. :P
- Sometimes **work gets stuck** and patches are not applied -for a number of reasons.
- **Patches** are **sometimes ignored**.
- Don’t want to miss patches **from occasional contributors.**
- Helpful to follow up on all patches sent to the linux-hardening list, so we can carry them on **our -next trees** when needed.
- https://patchwork.kernel.org/project/linux-hardening/

Issue tracker

- Issues show up while addressing other… **issues**.
- Sound familiar? :)
- Sometimes we need to **document** stuff before it’s included in the official documentation.
- We have **good first issues**. :)
- https://github.com/KSPP/linux/issues

- “I pulled and then immediately unpulled again.”
  - Linus Torvalds.

- "There is never too much information you can put in a merge commit. **Put what you ate on breakfast.** Put everything in there."
  - David Miller. #netconf2019

- Linus doesn’t care what you had for breakfast. :/

- Linus doesn’t care what you had for breakfast. :/
- But Dave does. Thanks Dave. :)

Coverity

- Not actually for kernel hardening, but it’s a good tool for new people.
- Public.
- Should be named linux-next-daily-scan.
- Daily scans for linux-next.
- Kees runs daily builds on his beefy machine.
- **Good place to start for newcomers**. Send a request for access. :)
- https://scan.coverity.com/projects/linux-next-weekly-scan/

Coccinelle

- We use it frequently.
- Not a magical solution for all we need to fix or change.
- **Code still should be audited**. :)
- https://coccinelle.gitlabpages.inria.fr/website/

Coccinelle

treewide: Replace zero-length arrays with flexible-array members

```
There is a regular need in the kernel to provide a way to declare
having a dynamically sized set of trailing elements in a structure.
Kernel code should always use “flexible array members”[1] for these
cases. The older style of one-element or zero-length arrays should
no longer be used[2].

This code was transformed with the help of Coccinelle:
(next-20220214$ spatch --jobs $(getconf _NPROCESSORS_ONLN) --sp-file script.co

@@
identifier S, member, array;
type T1, T2;
@@

struct S {
  ...
  T1 member;
  T2 array[
- 0
  ];
};
```

- https://git.kernel.org/linus/5224f79096170bf7b92cc8fe42a12f44b91e5f62

Coccinelle

- Potential struct_size() transformations. All should be audited.

```
@r1@
identifier VAR, ELEMENT;
expression COUNT, SOMETHING;
identifier id;
identifier id2;
@@

(
* sizeof(*VAR) + (COUNT * sizeof(*VAR->ELEMENT))
|
* sizeof(*VAR) + (COUNT * sizeof(VAR->ELEMENT[0]))
|
* sizeof(*VAR) + (COUNT * sizeof(struct id))
|
* sizeof(struct id) + (COUNT * sizeof(*VAR->ELEMENT))
|
* sizeof(struct id) + (COUNT * sizeof(VAR->ELEMENT[0]))
|
* sizeof(SOMETHING) + (COUNT * sizeof(ELEMENT))
|
* sizeof(SOMETHING) + (COUNT * sizeof(struct id))
|
* sizeof(struct id2) + (COUNT * sizeof(struct id))
)
```

Kernel Test Robot

- Build-tests for **multiple** archs and configurations. GCC and Clang.
- Results usually **within 24 hours**.
- **Need to ask** for your tree to be added to their test suite.
- Private and public reports. Depending on your preferences.
- Our test reports are **publicly available** on LKML.
- For complex changes I usually include a **Build-tested-by** tag with a link to the results.
- Kernel Test Robot <lkp@intel.com>

Kernel Test Robot

```
From: kernel test robot <lkp@intel.com>
To: "Gustavo A. R. Silva" <gustavoars@kernel.org>
Cc: LKML <linux-kernel@vger.kernel.org>
Subject: [gustavoars:for-next/kspp] BUILD SUCCESS 0cf2b91d74
Date: Sat, 14 May 2022 19:30:55 +0800   [thread overview]
Message-ID: <627f92ef.upN0G+bCpHComWxr%lkp@intel.com> (raw)

tree/branch: https://git.kernel.org/pub/scm/linux/kernel/git
branch HEAD: 0cf2b91d74b7ec0e971dcd00de875e2d04b56350  Merge

elapsed time: 13884m

configs tested: 153
configs skipped: 3

The following configs have been built successfully.
More configs may be tested in the coming days.

gcc tested configs:
arm                              allmodconfig
arm                                 defconfig
arm                              allyesconfig
arm64                               defconfig
arm64                            allyesconfig
i386                          randconfig-c001
mips                  randconfig-c004-20220505
powerpc                      ep8248e_defconfig
nios2                            allyesconfig
arm                        eseries_pxa_defconfig
sh                              se7780_defconfig
powerpc                        pq2fads_defconfig
m68k                             hp300_defconfig
ia64                             tiger_defconfig
m68k                             atari_defconfig
ia64                             tiger_defconfig
m68k                             atari_defconfig
```

```
clang tested configs:
x86_64                           randconfig-c007
i386                             randconfig-c001
powerpc                 randconfig-c003-20220505
riscv                   randconfig-c006-20220505
arm                     randconfig-c002-20220505
arm                            palmz72_defconfig
powerpc                      mvme5100_defconfig
powerpc                            g5_defconfig
arm                             ep93xx_defconfig
mips                         maltaaprp_defconfig
powerpc                    mpc836x_rdk_defconfig
arm                          mainstone_defconfig
powerpc                    mpc832x_rdb_defconfig
powerpc                  mpc834x_itxgp_defconfig
arm                               dove_defconfig
arm                         pxa255-idp_defconfig
mips                        cu1000-neo_defconfig
mips                              mtx1_defconfig
mips                         bmips_stb_defconfig
x86_64                           randconfig-a005
x86_64                           randconfig-a003
x86_64                           randconfig-a001
i386                             randconfig-a002
i386                             randconfig-a006
i386                             randconfig-a004
x86_64                           randconfig-a012
x86_64                           randconfig-a014
x86_64                           randconfig-a016
i386                             randconfig-a013
i386                             randconfig-a011
i386                             randconfig-a015
hexagon                 randconfig-r045-20220501
hexagon                 randconfig-r041-20220501
hexagon                 randconfig-r045-20220502
riscv                   randconfig-r042-20220502
hexagon                 randconfig-r041-20220502
```

- https://lore.kernel.org/lkml/627f92ef.upN0G+bCpHComWxr%25lkp@intel.com/

IRC channels and wiki

- Wanna hang out? :)
- #linux-hardening
- #clangbuiltlinux
- Libera.Chat
- https://kernsec.org/wiki/index.php/Kernel_Self_Protection_Project

What does it take to harden the kernel?

What does it take to harden the kernel?

- Sweat and blood!

What does it take to harden the kernel?

- Sweat and blood!
- Really!

What does it take to harden the kernel?

- KSPP is about hardening the Linux kernel.
- The goals are big.
- It’s usually not as glamorous as people would think.
- Auditing code is exhausting and time consuming.
- **We need to develop some strategies.**
- Make the compiler an ally.

What does it take to harden the kernel?

- Enabling **compiler options** is an important step forward.
- Provide the compiler with **enough context**.
- Detect as many problems as possible **at build-time**.
- Enabling compiler options is not that straightforward.
- **Upstream has its implications.** Who would’ve thought? :p
- Need to solve both **technical and political problems**.

What does it take to harden the kernel?

- **Political** issues tend to **delay** the work.
- Some people really dislike some changes.
- We need to convince people.
- People have different opinions about security changes across the whole kernel tree.
- Fortunately, some people really support the project.

What does it take to harden the kernel?

- A lot of **middle steps** need to land in order to complete important work.
- Clean ups and mechanical changes.
- Some are easy to implement, but hard to have them **applied upstream**.
- Maintainers usually don’t like a mechanical change happen external to their tree.
- **Avoid friction**.

Enabling compiler options

Enabling compiler options

- Why is it a **complex** task?
- Usually **tons** of warnings. :)
- **“The noisy thing.”**
- Some actual bugs, some false positives.
- Both are **worth** resolving.
- Some of those warnings lead us to find **corner cases** in both **kernel code** and **the compiler**.

Enabling compiler options

- Maybe we need to **change the narrative** a little bit.
- Small (although not simple) task are usually seen as noisy and code churn.
- Accomplishing important things require to **pay attention to small details, first**.
- The 99-1 rule. 99% **perspiration**/frustration, 1% inspiration/**innovation**.
- Improving the **quality and maintainability** of the code allows for trying to implement more complex stuff.

Work in progress and some accomplishments

- struct_group()
- Flexible array transformations.
- -Warray-bounds
- memcpy() hardening and the compound effect.

struct_group()

- Wrap a set of declarations in a mirrored struct.
- Group adjacent members in a struct to be accessed together. Usually through memcpy() or memset().
- Update to FORTIFY_SOURCE caused some memcpy() and memset() warnings when accessing multiple adjacent members of a structure at once.
- The flexibility of the C language. ;)

FORTIFY_SOURCE

- Uses the compiler's __builtin_object_size().
- Determine the available size at a target address based on the compile-time known structure layout details.
- Operates in two modes:
  - Outer bounds: __builtin_object_size(object, 0)
  - Inner bounds: __builtin_object_size(object, 1)
- More details: commit f68f2ff91512

__builtin_object_size()

- memcpy() under CONFIG_FORTIFY_SOURCE uses __builtin_object_size(object, 1)

```
struct object {
            u16 scalar1;    /* 2 bytes */
            char array[6];  /* 6 bytes */
            u64 scalar2;    /* 8 bytes */
            u32 scalar3;    /* 4 bytes */
            u32 scalar4;    /* 4 bytes */
} instance;

__builtin_object_size(instance.array, 0) == 22
__builtin_object_size(instance.array, 1) == 6

memcpy(&instance.array, source, 22); /* compile-time error! */
```

__struct_group()

- Create a mirrored named and anonymous struct.
- Commit 50d7bd38c3aa

```
 * Used to create an anonymous union of two structs with identical layout
 * and size: one anonymous and one named. The former's members can be used
 * normally without sub-struct naming, and the latter can be used to
 * reason about the start, end, and size of the group of struct members.
 * The named struct can also be explicitly tagged for layer reuse, as well
 * as both having struct attributes appended.
 */
#define __struct_group(TAG, NAME, ATTRS, MEMBERS...) \
        union { \
                struct { MEMBERS } ATTRS; \
                struct TAG { MEMBERS } ATTRS NAME; \
        }
```

struct_group()

- Group *entries* and *mac* into *sectors* and then do the memset(). (commit f069c7ab6cfb)

```
diff --git a/drivers/md/dm-integrity.c b/drivers/md/dm-integrity.c
index 7af242de3202e..eb4b5e52bd6ff 100644
--- a/drivers/md/dm-integrity.c
+++ b/drivers/md/dm-integrity.c
@@ -121,8 +121,10 @@ struct journal_entry {
 #define JOURNAL_MAC_SIZE               (JOURNAL_MAC_PER_SECTOR * JOURNAL_BLOCK_SECTORS)

 struct journal_sector {
-       __u8 entries[JOURNAL_SECTOR_DATA - JOURNAL_MAC_PER_SECTOR];
-       __u8 mac[JOURNAL_MAC_PER_SECTOR];
+       struct_group(sectors,
+               __u8 entries[JOURNAL_SECTOR_DATA - JOURNAL_MAC_PER_SECTOR];
+               __u8 mac[JOURNAL_MAC_PER_SECTOR];
+       );
        commit_id_t commit_id;
 };

@@ -2870,7 +2872,8 @@ static void init_journal(struct dm_integrity_c *ic, unsigned start_section,
                wraparound_section(ic, &i);
                for (j = 0; j < ic->journal_section_sectors; j++) {
                        struct journal_sector *js = access_journal(ic, i, j);
-                       memset(&js->entries, 0, JOURNAL_SECTOR_DATA);
+                       BUILD_BUG_ON(sizeof(js->sectors) != JOURNAL_SECTOR_DATA);
+                       memset(&js->sectors, 0, sizeof(js->sectors));
                        js->commit_id = dm_integrity_commit_id(ic, i, j, commit_seq);
```

Before struct_group()

- Addressing some -Warray-bounds warnings before struct_group() (commit 606636dcbdbb)

```
@@ -1176,12 +1180,14 @@ int intel_svm_page_response(struct device *dev,
                desc.qw1 = QI_PGRP_IDX(prm->grpid) | QI_PGRP_LPIG(last_page);
                desc.qw2 = 0;
                desc.qw3 = 0;
-               if (private_present)
-                       memcpy(&desc.qw2, prm->private_data,
-                              sizeof(prm->private_data));
-               else if (prm->private_data[0])
+
+               if (private_present) {
+                       desc.qw2 = prm->private_data[0];
+                       desc.qw3 = prm->private_data[1];
+               } else if (prm->private_data[0]) {
                        dmar_latency_update(iommu, DMAR_LATENCY_PRQ,
                                ktime_to_ns(ktime_get()) - prm->private_data[0]);
+               }
```

Before struct_group()

- Enclosing struct members into new structures *upiu_req* and *upiu_rsp* (commit 1352eec8c0da)

```
        /* DW 4-11 - Task request UPIU structure */
-       struct utp_upiu_header  req_header;
-       __be32                  input_param1;
-       __be32                  input_param2;
-       __be32                  input_param3;
-       __be32                  __reserved1[2];
+       struct {
+               struct utp_upiu_header  req_header;
+               __be32                  input_param1;
+               __be32                  input_param2;
+               __be32                  input_param3;
+               __be32                  __reserved1[2];
+       } upiu_req;

        /* DW 12-19 - Task Management Response UPIU struc
-       struct utp_upiu_header  rsp_header;
-       __be32                  output_param1;
-       __be32                  output_param2;
-       __be32                  __reserved2[3];
+       struct {
+               struct utp_upiu_header  rsp_header;
+               __be32                  output_param1;
+               __be32                  output_param2;
+               __be32                  __reserved2[3];
+       } upiu_rsp;
 };
```

```
0,7 @@ int ufshcd_exec_raw_upiu_cmd(struct ufs_hba *hba,
 treq.header.dword_0 = cpu_to_le32(UTP_REQ_DESC_INT_CMD);
 treq.header.dword_2 = cpu_to_le32(OCS_INVALID_COMMAND_STA

 memcpy(&treq.req_header, req_upiu, sizeof(*req_upiu));
 memcpy(&treq.upiu_req, req_upiu, sizeof(*req_upiu));

 err = __ufshcd_issue_tm_cmd(hba, &treq, tm_f);
 if (err == -ETIMEDOUT)
3,7 @@ int ufshcd_exec_raw_upiu_cmd(struct ufs_hba *hba,
         break;
 }

 memcpy(rsp_upiu, &treq.rsp_header, sizeof(*rsp_upiu));
 memcpy(rsp_upiu, &treq.upiu_rsp, sizeof(*rsp_upiu));
```

struct_group()

- struct_group()
- struct_group_attr()
- struct_group_tagged()
- More details in commit 50d7bd38c3aa

Flexible array transformations

```
struct ancient {
        size_t count;
        struct foo items[1];
};


struct old {
        size_t count;
        struct foo items[0];
};


struct modern {
        size_t count;
        struct foo items[];
};
```

Flexible array transformations

- 0-element was a GNU extension. 1-element was a hack.
- **Flexible array member** was introduced in C99.
- One-element arrays are particularly confusing and prone to error.
- zero-element and one-element arrays are **deprecated**.

  https://www.kernel.org/doc/html/latest/process/deprecated.html#zero-length-and-one-element-arrays
- These transformations are tricky and not that straightforward. I have introduced bugs (all fixed already :p) while doing flexible array transformations.

zero-length array

- sizeof(instance->items) == 0 (commits ab91c2a89f86, f2cd32a443da)

```
struct something {
        size_t count;
        struct foo items[0];
};

struct something *instance;
size_t size;

instance = kmalloc(struct_size(instance, items, n), GFP_KERNEL);
instance->count = n;

size = sizeof(instance->items) * instance->count; /* size == 0 */
memcpy(instance->items, source, size);
```

one-element array

- We have to remember to calculate *n - 1* when using the struct_size() helper.

```
struct something {
        size_t count;
        struct foo items[1];
};

struct something *instance;
size_t size;

instance = kmalloc(struct_size(instance, items, n - 1), GFP_KERNEL);
instance->count = n;

size = sizeof(instance->items) * instance->count;
memcpy(instance->items, source, size);
```

flexible array member

- Now, with the hardened memcpy(), we can omit the use of flex_array_size() in this case.

```
struct something {
        size_t count;
        struct foo items[];
};

struct something *instance;
size_t size;

instance = kmalloc(struct_size(instance, items, n), GFP_KERNEL);
instance->count = n;

memcpy(instance->items, source, flex_array_size(instance, items,
                                                instance->count));
```

Flexible arrays and trailing arrays

- Compilers treat all **trailing arrays as flexible arrays**.
- It **breaks FORTIFY_SOURCE** in that any struct with a fixed size trailing array will **receive no sanity checking.**
- It seems that there are a lot of legacy code “taking advantage” of that.
- -fstrict-flex-array?
- GCC and Clang bugs:
  https://gcc.gnu.org/bugzilla/show_bug.cgi?id=101419
  https://github.com/llvm/llvm-project/issues/55741

-Warray-bounds

- Enabled for GCC 11 and earlier. :)
- 153 more new warnings with GCC 12
- v5.18, GCC 11 to GCC 12:

  $ grep warning: fedora36.log | grep 'Warray-bounds' | wc -l

  153
- https://github.com/KSPP/linux/issues/190

-Warray-bounds

- It is finding bugs. :)
- https://lore.kernel.org/lkml/202204201117.F44DCF9@keescook/

```
I quickly went back through commits; here's a handful I found:

20314bacd2f9 ("staging: r8188eu: Fix PPPoE tag insertion on little endian systems")
dcf500065fab ("net: bnxt_ptp: fix compilation error")
5f7dc7d48c94 ("octeontx2-af: fix array bound error")
c7d58971dbea ("ALSA: mixart: Reduce size of mixart_timer_notify")
b3f1dd52c991 ("ARM: vexpress/spc: Avoid negative array index when !SMP")
a2151490cc6c ("drm/dp: Fix OOB read when handling Post Cursor2 register")
d4da1f27396f ("drm/dp: Fix off-by-one in register cache size")
47307c31d90a ("crypto: octeontx2 - Avoid stack variable overflow")
a6501e4b380f ("eeprom: at25: Restore missing allocation")
33ce7f2f95ca ("drm/imx: imx-ldb: fix out of bounds array access warning")
f051ae4f6c73 ("Input: cyapa_gen6 - fix out-of-bounds stack access")
f3217d6f2f7a ("firmware: xilinx: fix out-of-bounds access")
8a03447dd311 ("rtw88: fix subscript above array bounds compiler warning")
ad82a928eb58 ("s390/perf: fix gcc 8 array-bounds warning")
6038aa532a22 ("nvme: target: fix buffer overflow")
50a0d71a5d20 ("cros_ec: fix nul-termination for firmware build info")
43d15c201313 ("staging: rtl8822be: Keep array subscript no lower than zero")
```

Other work

- -Wstringop-overflow making progress.

  https://git.kernel.org/linus/a3a8b54b4f1a
- -Wimplicit-fallhrough for Clang is now enabled. We had almost 40,000 warnings.

  https://github.com/KSPP/linux/issues/115
- -Wcast-function-type is now enabled.

  https://git.kernel.org/linus/01367e86e90

memcpy() hardening and the compound effect.

- https://outflux.net/slides/2022/lca/

```
__FORTIFY_INLINE void *memcpy(void *dst, const void *src, size_t size)
{
        size_t dst_size = __builtin_object_size(dst, 1);
        size_t src_size = __builtin_object_size(src, 1);

        if (__builtin_constant_p(size)) {       /* Compile-time */
                if (dst_size < size)
                        __write_overflow();
                if (src_size < size)
                        __read_overflow2();
        }
        if (dst_size < size || src_size < size)
                fortify_panic(__func__);        /* Run-time */
        return __underlying_memcpy(dst, src, size);
}
```

How you can help. :)

How you can help. :)

- Doing flexible array transformations, of course. :)

  https://github.com/KSPP/linux/issues/79

- Issue 79 contains a list with hundreds of patches that have landed in mainline. They can be used as examples.
- Audit code and use the helpers available, when possible: struct_size(), struct_group(), flex_array_size(), size_add(), size_mul(), etc. See commit e1be43d9b5d0
- Take a look at the issue tracker. We have issues for everybody. :)

  https://github.com/KSPP/linux/issues/

one-element arrays to flexible arrays

- Find **uses of sizeof** on the involved struct-with-one-element-array or on the **type** of the one-element array itself.
- Find **the *n – 1* pattern** and change it to just ***n***. If you don't find this pattern then something else may be going on, and you need to **carefully verify that the size for the allocation and the iteration over the array are correct**. Otherwise, chances are you just found a bug (usually an off-by-one error).
- Look for any iteration over the array and verify it is still **within the boundaries**. Usually in the form of a for loop. Use diffoscope before/after the changes to check the binary.
- **CC-me:** gustavoars@kernel.org

one-element arrays to flexible arrays

- The ideal scenario. (commit c72a826829cc)

```
diff --git a/fs/nfs/filelayout/filelayout.h b/fs/nfs/filelayout/filelayo
index 79323b5dab0cb..aed0748fd6ec7 100644
--- a/fs/nfs/filelayout/filelayout.h
+++ b/fs/nfs/filelayout/filelayout.h
@@ -51,7 +51,7 @@ struct nfs4_file_layout_dsaddr {
        u32                             stripe_count;
        u8                              *stripe_indices;
        u32                             ds_num;
-       struct nfs4_pnfs_ds             *ds_list[1];
+       struct nfs4_pnfs_ds             *ds_list[];
 };

 struct nfs4_filelayout_segment {

diff --git a/fs/nfs/filelayout/filelayoutdev.c b/fs/nfs/filelayout/filel
index 86c3f7e69ec42..acf4b88889dc3 100644
--- a/fs/nfs/filelayout/filelayoutdev.c
+++ b/fs/nfs/filelayout/filelayoutdev.c
@@ -136,9 +136,7 @@ nfs4_fl_alloc_deviceid_node(struct nfs_server *serve
                goto out_err_free_stripe_indices;
        }

-       dsaddr = kzalloc(sizeof(*dsaddr) +
-                        (sizeof(struct nfs4_pnfs_ds *) * (num - 1)),
-                        gfp_flags);
+       dsaddr = kzalloc(struct_size(dsaddr, ds_list, num), gfp_flags);
        if (!dsaddr)
                goto out_err_free_stripe_indices;
```

Conclusions

The best outcome

- In regards to the technical portion of this work, we want to achieve having a **robust and hardened kernel** with **secure core infrastructure and safer APIs**, that allow us to both **eliminate entire bug classes and methods of exploitation** in the upstream Linux kernel. That definitely would be the best outcome.

Political work

- Hopefully the social and **political work** we've been doing all this time will **make it easier** to introduce more changes that **improve the security of the kernel** and, at the same time, **benefit new people** that want to collaborate with us by helping them **navigate with ease** the, sometimes, **wild waters** of the Linux kernel community.

A commit at a time

- Change in the kernel, especially in terms of security, is **an evolutionary process**. It is slow and demands a lot of **patience**. There is still more work than we can get done. We always welcome people who can help out. Companies participating in any ecosystem that's based on Linux need to really consider funding projects that improve the overall security of the kernel. This is an effort that is **driving change a commit at a time** and, that **benefits billions of people around the world**, including of course, users and customers of tech companies of all sizes.

  https://security.googleblog.com/2021/08/linux-kernel-security-done-right.html

Thank you! :)

Gustavo A. R. Silva
gustavoars@kernel.org
@embeddedgus