April 27, 2021

A Full Disclosure of the Case Study of the “Hypocrite Commits” Paper

Note on consent: We apologize for the delay in getting this information to the community. We wanted to ensure that we had the consent of all concerned before making this information public. We just secured consent from all who responded to our patches this morning and can now publish this information.

This document provides the details of the case study related to the Linux community for the “hypocrite commits" paper (https://github.com/QiushiWu/QiushiWu.github.io/blob/main/papers/OpenSourceInsecurity.pdf). We have chosen to withdraw this paper from publication (https://www-users.cs.umn.edu/~kjlu/papers/withdrawal-letter.pdf).

This paper is about studying the feasibility of introducing vulnerabilities into open source projects through a minor patch that (1) does not add or change any functionalities, (2) does not contain a vulnerability by itself, and (3) introduces only a small code change (e.g., less than 5 lines of code change). In particular, it focuses on studying causes of the feasibility, factors affecting the feasibility, and suggestions for mitigating the feasibility. The vulnerability-introducing method proposed in the paper is about how to introduce a vulnerability condition (not a vulnerability by itself) through a minor patch in such a way that it turns an “immature vulnerability” into a real one without being detected during the patch review process. To show that this was a real problem, we carried out a proof-of-concept case study by starting the process of proposing 3 "hypocrite patches" (patches introducing a vulnerability condition, each has 1-4 lines of code change, thus is called “minor patch”). The patches by themselves do not contain vulnerabilities. These patches were proposed in a way that would prevent them from ever being introduced into the kernel. All of the "hypocrite patches" were aborted before they could move forward, as was the plan and as can be seen in the following details.

In the following we will show two parts: (1) the message log of our disclosure of the findings to the community, and (2) the patches we submitted. By showing the details of the patches and the exchange of messages, we wish to help the community to confirm that the buggy patches were “stopped” during message exchanges and not merged into the actual Linux code.

No other interactions with the Linux Kernel team has involved intentional deception or intentionally misleading or bad patches. This misguided behavior on our part was limited to the patches described and clarified in this document.

Part 1: Disclosure of Findings Before Paper Submission

Related emails (the full exchange is also included in Appendix A)
1.The original message from Qiushi Wu. It was not included in LKML due to the email not being in a plaintext form, but its copy can be found in Greg's reply and Appendix A.
2.https://lkml.org/lkml/2020/8/27/1197
3.https://lkml.org/lkml/2020/8/28/51
4.https://lkml.org/lkml/2020/8/28/139
5.https://lkml.org/lkml/2020/8/28/167
6.https://lkml.org/lkml/2020/8/28/221

Part 2: All Patches in This Work

acostag.ubuntu@gmail.com and jameslouisebond@gmail.com are all anonymous email addresses used in the study. There are in total 5 patches sent out through these two emails: 3 are buggy, and 2 are valid, as listed below. The fifth one is not part of the study. Here are all the patches and their corresponding URL links.

#	URLs (the full exchanges are also included in Appendix B)
Patch 1	a.https://lore.kernel.org/lkml/20200821031209.21279-1-acostag.ubuntu@gmail.com/#t b.https://lore.kernel.org/lkml/20200828071931.GB28064@gondor.apana.org.au/ c.https://lore.kernel.org/lkml/CALhW5_QpsRCb73OCiOKC0xVSwuadz3BVSQg+r=T4AN+qCpSM0w@mail.gmail.com/ d.https://lore.kernel.org/lkml/20200829120409.GA22368@gondor.apana.org.au/
Patch 2	a.https://lore.kernel.org/lkml/20200809221453.10235-1-jameslouisebond@gmail.com/ b.https://lore.kernel.org/lkml/dd4ac5db-9ae7-fbf7-535b-c0d3fa098942@kernel.org/ c.https://lore.kernel.org/lkml/20200810075122.GA1531406@kroah.com/ d.https://lore.kernel.org/lkml/08df63cd-c4b9-c16b-2e27-5d86580eebf2@kernel.org/
Patch 3	a.https://lore.kernel.org/lkml/20200821034458.22472-1-acostag.ubuntu@gmail.com/ b.https://lore.kernel.org/lkml/20200821081449.GI5493@kadam/
Patch 4	a.https://lkml.org/lkml/2020/8/21/124 b.https://lkml.org/lkml/2020/8/27/614 c.https://lkml.org/lkml/2020/8/28/1046 d.https://lkml.org/lkml/2020/8/31/362

Patch 5	https://lore.kernel.org/lkml/20200804183650.4024-1-jameslouisebond@gmail.com/

- Patch 1 corresponds to Figure 11 (case 1) in the paper. This is not a buggy patch, but included in the paper for the demonstration of a representative case.
- Patch 2 corresponds to Figure 9 (case 2) in the paper. This is a buggy patch.
- Patch 3 corresponds to Figure 10 (case 3) in the paper. This is a buggy patch.
- Patch 4 is a buggy patch, but not included in the paper, because its concept is similar to the concept of patch 1 and patch 3.
- Patch 5 is a valid patch and is not part of the project, but it was sent out through one of the anonymous emails by accident.

Among these five patches, only the first one (valid) is merged into the kernel code. We choose to include patches 1, 2, and 3 in the paper as proof-of-concept to better demonstrate the practicality of representative cases. We hope the details confirm that **(1) none of these cases introduced actual bugs in the Linux code, and (2) we did not intend to introduce actual bugs.**

Patch 1 (Figure 11 in the paper). This is not a buggy patch. At first, we thought this patch was a "hypocrite commit" and submitted it to the community. However, after further checking, we confirmed that this patch is actually valid and would not introduce buggy code into the kernel, so we didn't stop the maintainer from continuing to apply the patch. Although this case is not buggy, error handling paths are often less taken care of, especially when the free function is custom. Sometimes even when the free function is commonly used like kfree, use-after-free is still introduced, and the patch is accepted and applied, such as https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=f187b6974f6df (not from us, applied in 2020). Also, involving concurrency would make it much stealthier. Therefore, we believe that such a case, using a pointer in error paths after functions like close, disable, release, and destroy, is practical. We included this one in the paper to better illustrate such a representative case. However, since this specific patch is not a buggy patch, the paper should have explained this clearly.

Patch 2 (Figure 9 in the paper). This is a buggy patch and not merged into the kernel code. In this case, during the study, we first ran Syzkaller+KMEMLEAK and found a memory leak. After we sent this patch to the maintainers, they pointed out that a similar (almost the same) patch (https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=84ecc2f6eb1cb12e6d44818f94fa49b50f06e6ac), from someone else, had been accepted, applied, and then reverted in 2019, because it introduced use-after-free. Note that we did not submit the 2019 patch, and we also didn't know that patch during the work. However, it clearly shows that such a buggy patch is “practical”, as it has been accepted and applied once, so we also included this case in the paper as a proof-of-concept. Since our patch was clearly dropped by maintainers, it never entered the Linux code.
This is the previously applied and reverted patch in 2019 (not from us):

https://lore.kernel.org/lkml/20190524080200.GA19609@kroah.com/

Patch 3 (Figure 10 in the paper). This is a buggy patch. This case was also stopped during the discussion and not merged into the kernel. The maintainer caught a coding issue and another buggy issue caused by list_add_tail. However, during the review, the maintainer did not capture the introduced use-after-free caused by the use of pointer "chdev" in dev_err(&chdev->dev..), which may have been released in the introduced put_device function. Since the buggy patch has issues pointed out by the maintainer, and we did not submit an “improved” patch, this buggy patch was stopped and not merged into the kernel.

Patch 4 (not included in the paper). This is a buggy patch and not merged into kernel code. In this case, during the review, the maintainer did not capture the vulnerability---"pdev" is put inside ck804xrom_cleanup, and thus a double-put bug would occur after calling this put function in the patch. The maintainer however pointed out another coding issue in the patch and the patched function. Since the maintainer has pointed out other issues, it is clear that this buggy patch will not be applied to Linux.

Patch 5. This is a valid patch that was intended to fix a refcount release problem mentioned in another project, which can also fit the description of function kobject_init_and_add---"If this function returns an error, kobject_put() must be called to properly clean up the memory associated with the object." (see https://elixir.bootlin.com/linux/v5.12/source/lib/kobject.c#L464) (Also see section 8 “Suggestions for avoiding DiEH” part of the paper https://github.com/QiushiWu/QiushiWu.github.io/blob/main/papers/hero.pdf).

We had configured the “git send-email” to the anonymous email (jameslouisebond@gmail.com) in the end of July 2020 and were preparing for the case study. During checking the source code of the Linux kernel, and preparing for finding the placement of hypocrite commits, we found this bug. Due to it being the same class of bugs as we previously identified and patched in a previous project (https://github.com/QiushiWu/QiushiWu.github.io/blob/main/papers/hero.pdf), we also sent the valid patch of this bug to Linux, however, forgot to change the anonymous email back to my generally used personal email. Therefore, patch 5 is not part of the “hypocrite commits” study or hypocrite commits. Due to the mistake, although this patch is not merged into the kernel code, we still include it here for an explanation.

Appendix A
Part 1:

1. The original message from Qiushi Wu. It was not included in LKML due to the email not being in a plaintext form, but its copy can also be found in Greg's reply.

Qiushi Wu <wu000273@umn.edu>
Thu, Aug 27, 2020, 12:34 PM
to LKML, Greg, David
Subject: Some questions about the patching process

Dear Linux maintainers:

I'm Qiushi Wu, a Ph.D. student from the secure and reliable systems research group at the University of Minnesota. Our group strives to improve the security and reliability of the Linux kernel and has contributed quite a number of patches. We appreciate the openness of the Linux community, but would also like to discuss some questions about the patching process. It would be great if you could let us know your thoughts.

We recently found that minor patches such as one fixing a memory leak may introduce more critical security bugs like double free. Sometimes the maintainers can capture the introduced security bugs, sometimes not. This is understandable because the introduced bugs can be subtle and hard to catch due to the complexity. We are more concerned when "bad" submitters intentionally and stealthily introduce such security bugs via seemingly good patches, as they indeed have a chance to get the actually bad patches accepted. This is not impossible because people have incentives---to plant a vulnerability in a targeted driver, to get bounty rewards, etc.

Based on our patching experience, we observe several things related to the risks.

1. Linux allows anyone to submit a patch because it is an open community.

2. Maintainers tend to not accept preventive patches, i.e., a patch for a bug that is not really there yet, but it can be likely formed in the future.

3. The patch review is mainly manual, so sometimes the introduced security bugs could be missed.

We would like to know how the Linux maintainers think about these risks. We would like to know if maintainers have some methods and tools (such as Smatch, Syzbot?) to mitigate these potential issues. We are happy to discuss these issues and hope our observations could raise some awareness of them.

Best regards,

Qiushi

2. https://lkml.org/lkml/2020/8/27/1197

Date Thu, 27 Aug 2020 20:27:30 +0200
From Greg Kroah-Hartman <>
Subject Re: Some questions about the patching process
share 0
On Thu, Aug 27, 2020 at 12:34:57PM -0500, Qiushi Wu wrote:
> Dear Linux maintainers:
>
> I'm Qiushi Wu, a Ph.D. student from the secure and reliable systems
> research group at the University of Minnesota. Our group strives to improve
> the security and reliability of the Linux kernel and has contributed quite
> a number of patches. We appreciate the openness of the Linux community, but
> would also like to discuss some questions about the patching process. It
> would be great if you could let us know your thoughts.
>
> We recently found that minor patches such as one fixing a memory leak may
> introduce more critical security bugs like double free. Sometimes the
> maintainers can capture the introduced security bugs, sometimes not. This
> is understandable because the introduced bugs can be subtle and hard to
> catch due to the complexity. We are more concerned when “bad” submitters
> intentionally and stealthily introduce such security bugs via seemingly
> good patches, as they indeed have a chance to get the actually bad patches
> accepted. This is not impossible because people have incentives---to plant
> a vulnerability in a targeted driver, to get bounty rewards, etc.
>
> Based on our patching experience, we observe several things related to the
> risks.
>
> 1. Linux allows anyone to submit a patch because it is an open community.
>
> 2. Maintainers tend to not accept preventive patches, i.e., a patch for a
> bug that is not really there yet, but it can be likely formed in the
> future.

How can you create a patch to prevent a bug that is not present?

But no, this is not true, look at all of the kernel hardening features
added to the kernel over the past 5+ years. Those are specifically to

help handle the problem when there are bugs in the kernel, so that "bad things" do not happen when they occur.

> 3. The patch review is mainly manual, so sometimes the introduced security
> bugs could be missed.

We are human, no development process can prevent this.

> We would like to know how the Linux maintainers think about these risks.

I think 2. is wrong, so it's not a risk.

And how is 1. a "risk"?

And of course, 3., humans, well, what can you do about them? :)

> We
> would like to know if maintainers have some methods and tools (such as
> Smatch, Syzbot?) to mitigate these potential issues. We are happy to
> discuss these issues and hope our observations could raise some awareness
> of them.

How do you "raise awareness" among a developer community that is 4000 people each year (1000 are new each year), consisting of 450+ different companies?

And yes, we have lots of tools, and run them all the time on all of our public trees constantly. And they fix things before they get merged and sent out to the rest of the world.

So what specific things are you wanting to discuss here?

thanks,

greg k-h

3. https://lkml.org/lkml/2020/8/28/51

Date Fri, 28 Aug 2020 08:20:42 +0200
From Greg Kroah-Hartman <>
Subject Re: Some questions about the patching process
share 0
On Thu, Aug 27, 2020 at 02:17:20PM -0500, Qiushi Wu wrote:

> Hi Greg,
> Thanks for your response!

<snip>

You responded in html format which got rejected by the public list,
please resend in text-only and I will be glad to reply.

thanks,

greg k-h

4. https://lkml.org/lkml/2020/8/28/139

From Qiushi Wu <>
Date Fri, 28 Aug 2020 02:59:25 -0500
Subject Re: Some questions about the patching process
share 0
Hi Greg,
Thanks for your response!

> You responded in html format which got rejected by the public list,
> please resend in text-only and I will be glad to reply.
>
Sorry about this!

> > 1. Linux allows anyone to submit a patch because it is an open community.
> >
> And how is 1. a "risk"?

We are assuming the possibility of potential malicious commit contributors
and want to reduce the risk of accepting vulnerable patches from them.

> > We would like to know if maintainers have some methods and tools (such as
> > Smatch, Syzbot?) to mitigate these potential issues. We are happy to
> > discuss these issues and hope our observations could raise some awareness
> > of them.
>
> How do you "raise awareness" among a developer community that is 4000
> people each year (1000 are new each year), consisting of 450+ different
> companies?

Yes, this is a problem. Maybe people can summarize and pubic some security coding guidelines for different modules of the kernel, or recommend maintainers to use some bug detection tools to test the patches.

> And yes, we have lots of tools, and run them all the time on all of our
> public trees constantly. And they fix things before they get merged and
> sent out to the rest of the world.
>
> So what specific things are you wanting to discuss here?

Specifically, we are curious about what kind of tools maintainers are often used to test potential bugs or vulnerabilities? Also, can these tools have a high coverage rate to test corner cases like error-paths, indirect calls, concurrency issues, etc?

Thanks again for your patience and reply :),
Best regards,
Qiushi

5. https://lkml.org/lkml/2020/8/28/167

Date Fri, 28 Aug 2020 10:26:08 +0200
From Greg Kroah-Hartman <>
Subject Re: Some questions about the patching process
share 0
On Fri, Aug 28, 2020 at 02:59:25AM -0500, Qiushi Wu wrote:
> Hi Greg,
> Thanks for your response!
>
> > You responded in html format which got rejected by the public list,
> > please resend in text-only and I will be glad to reply.
> >
> Sorry about this!
>
>
> > > 1. Linux allows anyone to submit a patch because it is an open community.
> > >
> > And how is 1. a "risk"?
>
> We are assuming the possibility of potential malicious commit contributors
> and want to reduce the risk of accepting vulnerable patches from them.

No, you are thinking about this all wrong.

ALL contributors make mistakes, you should not be treating anyone different from anyone else. I think I probably have contributed more bugs than many contributors, does that make me a "malicious" contributor? Or just someone who contributes a lot?

So checking on patches needs to be done for everyone, right?

We have an idea of "trust" in kernel development, it's how we work so well. I don't trust people not that they will always get things "correct", but rather that they will be around to fix it when they get it "wrong", as everyone makes mistakes, we are all human.

So we trust people who we accept pull requests from, we don't review their contributions because we trust that they did, and again, they will fix it when it goes wrong.

> > > We would like to know if maintainers have some methods and tools (such as
> > > Smatch, Syzbot?) to mitigate these potential issues. We are happy to
> > > discuss these issues and hope our observations could raise some awareness
> > > of them.
> >
> > How do you "raise awareness" among a developer community that is 4000
> > people each year (1000 are new each year), consisting of 450+ different
> > companies?
>
> Yes, this is a problem. Maybe people can summarize and pubic some security
> coding guidelines for different modules of the kernel, or recommend maintainers
> to use some bug detection tools to test the patches.

We do both today quite well, why do you think this is not the case?

> > And yes, we have lots of tools, and run them all the time on all of our
> > public trees constantly. And they fix things before they get merged and
> > sent out to the rest of the world.
> >
> > So what specific things are you wanting to discuss here?
>
> Specifically, we are curious about what kind of tools maintainers are often
> used to test potential bugs or vulnerabilities?

We use lots, everything we do is in the open, I suggest doing some research first please.

> Also, can these tools have a
> high coverage rate to test corner cases like error-paths, indirect calls,
> concurrency issues, etc?

Since when does code coverage actually matter as a viable metric?

Look at the tools we use, again, it's all in the open, and tell us what
we could be doing differently by offering to help us implement those
tools into our workflows. That would be the best way to contribute
here, don't you agree?

thanks,

greg k-h

6. https://lkml.org/lkml/2020/8/28/221

From Qiushi Wu <>
Date Fri, 28 Aug 2020 04:11:46 -0500
Subject Re: Some questions about the patching process
share 0
On Fri, Aug 28, 2020 at 3:25 AM Greg Kroah-Hartman
<gregkh@linuxfoundation.org> wrote:
>
> On Fri, Aug 28, 2020 at 02:59:25AM -0500, Qiushi Wu wrote:
> > Hi Greg,
> > Thanks for your response!
> >
> > > You responded in html format which got rejected by the public list,
> > > please resend in text-only and I will be glad to reply.
> > >
> > Sorry about this!
> >
> >
> > > > 1. Linux allows anyone to submit a patch because it is an open community.
> > > >
> > > And how is 1. a "risk"?
> >
> > We are assuming the possibility of potential malicious commit contributors
> > and want to reduce the risk of accepting vulnerable patches from them.
>
> No, you are thinking about this all wrong.
>

> ALL contributors make mistakes, you should not be treating anyone
> different from anyone else. I think I probably have contributed more
> bugs than many contributors, does that make me a "malicious"
> contributor? Or just someone who contributes a lot?

Sorry for my confusion! I don't mean to say that our kernel
contributors are 'malicious' or some similar, and previously I have also
made mistakes and send buggy code into the kernel accidentally.
Also, we are trying to summarize the methods to efficiently auditing
patches to prevent potential issues in the patch.

> So checking on patches needs to be done for everyone, right?
>
> We have an idea of "trust" in kernel development, it's how we work so
> well. I don't trust people not that they will always get things
> "correct", but rather that they will be around to fix it when they get
> it "wrong", as everyone makes mistakes, we are all human.
>
> So we trust people who we accept pull requests from, we don't review
> their contributions because we trust that they did, and again, they will
> fix it when it goes wrong.

agree :)

> We use lots, everything we do is in the open, I suggest doing some
> research first please.
> > Also, can these tools have a
> > high coverage rate to test corner cases like error-paths, indirect calls,
> > concurrency issues, etc?
>
> Since when does code coverage actually matter as a viable metric?
>
> Look at the tools we use, again, it's all in the open, and tell us what
> we could be doing differently by offering to help us implement those
> tools into our workflows. That would be the best way to contribute
> here, don't you agree?
Okay, I see.

Thanks again for your patient reply, and apologies for my confusing description.

Best regards,
Qiushi

Appendix B

Patch 1

1) https://lore.kernel.org/lkml/20200821031209.21279-1-acostag.ubuntu@gmail.com/#t

```
From: George Acosta <acostag.ubuntu@gmail.com>
To: acostag.ubuntu@gmail.com
        "David S. Miller" <davem@davemloft.net>,
        Christophe JAILLET <christophe.jaillet@wanadoo.fr>,
        "Gustavo A. R. Silva" <gustavo@embeddedor.com>,
        Phani Kiran Hemadri <phemadri@marvell.com>,
        linux-crypto@vger.kernel.org, linux-kernel@vger.kernel.org
Subject: [PATCH] crypto: cavium/nitrox: add an error message to explain the failure of
pci_request_mem_regions
Date: Thu, 20 Aug 2020 22:12:08 -0500
Message-ID: <20200821031209.21279-1-acostag.ubuntu@gmail.com> (raw)

Provide an error message for users when pci_request_mem_regions failed.

Signed-off-by: George Acosta <acostag.ubuntu@gmail.com>
---
 drivers/crypto/cavium/nitrox/nitrox_main.c | 1 +
 1 file changed, 1 insertion(+)

diff --git a/drivers/crypto/cavium/nitrox/nitrox_main.c b/drivers/crypto/cavium/nitrox/nitrox_main.c
index cee2a2713038..9d14be97e381 100644
--- a/drivers/crypto/cavium/nitrox/nitrox_main.c
+++ b/drivers/crypto/cavium/nitrox/nitrox_main.c
@@ -451,6 +451,7 @@ static int nitrox_probe(struct pci_dev *pdev,
        err = pci_request_mem_regions(pdev, nitrox_driver_name);
        if (err) {
                pci_disable_device(pdev);
+               dev_err(&pdev->dev, "Failed to request mem regions!\n");
                return err;
        }
        pci_set_master(pdev);
--
2.17.1
```

2) https://lore.kernel.org/lkml/20200828071931.GB28064@gondor.apana.org.au/

```
On Thu, Aug 20, 2020 at 10:12:08PM -0500, George Acosta wrote:
> Provide an error message for users when pci_request_mem_regions failed.
>
> Signed-off-by: George Acosta <acostag.ubuntu@gmail.com>
> ---
```

> drivers/crypto/cavium/nitrox/nitrox_main.c | 1 +
> 1 file changed, 1 insertion(+)

Patch applied. Thanks.
--
Email: Herbert Xu <herbert@gondor.apana.org.au>
Home Page: http://gondor.apana.org.au/~herbert/
PGP Key: http://gondor.apana.org.au/~herbert/pubkey.txt

3) https://lore.kernel.org/lkml/CALhW5_QpsRCb73OCiOKC0xVSwuadz3BVSQg+r=T4AN+qCpSM0w@mail.gmail.com/

I just noticed a small potential issue about the calling sequence of
pci_disable_device and dev_err.
Do you think it will be better to call dev_err before we call
pci_disable_device(pdev) , or the order here does not matter?

On Fri, Aug 28, 2020 at 2:19 AM Herbert Xu <herbert@gondor.apana.org.au> wrote:
>
> On Thu, Aug 20, 2020 at 10:12:08PM -0500, George Acosta wrote:
> > Provide an error message for users when pci_request_mem_regions failed.
> >
> > Signed-off-by: George Acosta <acostag.ubuntu@gmail.com>
> > ---
> > drivers/crypto/cavium/nitrox/nitrox_main.c | 1 +
> > 1 file changed, 1 insertion(+)
>
> Patch applied. Thanks.
> --
> Email: Herbert Xu <herbert@gondor.apana.org.au>
> Home Page: http://gondor.apana.org.au/~herbert/
> PGP Key: http://gondor.apana.org.au/~herbert/pubkey.txt

Thanks,
George

4) https://lore.kernel.org/lkml/20200829120409.GA22368@gondor.apana.org.au/
On Fri, Aug 28, 2020 at 08:53:36PM -0500, George Acosta wrote:

> I just noticed a small potential issue about the calling sequence of

> pci_disable_device and dev_err.
> Do you think it will be better to call dev_err before we call
> pci_disable_device(pdev) , or the order here does not matter?

I don't think it matters.

Thanks,
--
Email: Herbert Xu <herbert@gondor.apana.org.au>
Home Page: http://gondor.apana.org.au/~herbert/
PGP Key: http://gondor.apana.org.au/~herbert/pubkey.txt

Patch 2

1) https://lore.kernel.org/lkml/20200809221453.10235-1-jameslouisebond@gmail.com/

James Bond <jameslouisebond@gmail.com>
Sun, Aug 9, 2020, 5:14 PM
to me, Greg, Jiri, Andrew, Kees, Michel, Vlastimil, Denis, linux-kernel

Syzkaller find a memory leak in con_insert_unipair:

```
BUG: memory leak
unreferenced object 0xffff88804893d100 (size 256):
comm "syz-executor.3", pid 16154, jiffies 4295043307 (age 2392.340s)
hex dump (first 32 bytes):
80 af 88 4e 80 88 ff ff 00 a8 88 4e 80 88 ff ff  ...N.......N....
80 ad 88 4e 80 88 ff ff 00 aa 88 4e 80 88 ff ff  ...N.......N....
backtrace:
[<00000000f76ff1de>] kmalloc include/linux/slab.h:555 [inline]
[<00000000f76ff1de>] kmalloc_array include/linux/slab.h:596 [inline]
[<00000000f76ff1de>] con_insert_unipair+0x9e/0x1a0 drivers/tty/vt/consolemap.c:482
[<000000002f1ad7da>] con_set_unimap+0x244/0x2a0 drivers/tty/vt/consolemap.c:595
[<0000000046ccb106>] do_unimap_ioctl drivers/tty/vt/vt_ioctl.c:297 [inline]
[<0000000046ccb106>] vt_ioctl+0x863/0x12f0 drivers/tty/vt/vt_ioctl.c:1018
[<00000000db1577ff>] tty_ioctl+0x4cd/0xa30 drivers/tty/tty_io.c:2656
[<00000000e5cdf5ed>] vfs_ioctl fs/ioctl.c:48 [inline]
[<00000000e5cdf5ed>] ksys_ioctl+0xa6/0xd0 fs/ioctl.c:753
[<00000000fb4aa12c>] __do_sys_ioctl fs/ioctl.c:762 [inline]
[<00000000fb4aa12c>] __se_sys_ioctl fs/ioctl.c:760 [inline]
[<00000000fb4aa12c>] __x64_sys_ioctl+0x1a/0x20 fs/ioctl.c:760
[<00000000f561f260>] do_syscall_64+0x4c/0xe0 arch/x86/entry/common.c:384
[<0000000056206928>] entry_SYSCALL_64_after_hwframe+0x44/0xa9
BUG: leak checking failed
```

To fix this issue, we need to release the pointer p1 when the call of
the function kmalloc_array fail.

Signed-off-by: James Bond <jameslouisebond@gmail.com>
---
 drivers/tty/vt/consolemap.c | 5 ++++-
 1 file changed, 4 insertions(+), 1 deletion(-)

diff --git a/drivers/tty/vt/consolemap.c b/drivers/tty/vt/consolemap.c
index 5947b54d92be..1e8d06c32ca1 100644
--- a/drivers/tty/vt/consolemap.c
+++ b/drivers/tty/vt/consolemap.c
@@ -489,7 +489,10 @@ con_insert_unipair(struct uni_pagedir *p, u_short unicode, u_short
fontpos)
        p2 = p1[n = (unicode >> 6) & 0x1f];
        if (!p2) {
                p2 = p1[n] = kmalloc_array(64, sizeof(u16), GFP_KERNEL);
-               if (!p2) return -ENOMEM;
+   if (!p2) {
+   kfree(p1);
+   return -ENOMEM;
+   }
                memset(p2, 0xff, 64*sizeof(u16)); /* No glyphs for the characters (yet) */
        }

--
2.17.1

2) https://lore.kernel.org/lkml/dd4ac5db-9ae7-fbf7-535b-c0d3fa098942@kernel.org/
Jiri Slaby jirislaby@kernel.org via gmail.com
Aug 10, 2020, 12:16 AM
to Ben, me, Greg, Andrew, Kees, Michel, Vlastimil, Denis, linux-kernel

This reminds me of:
commit 15b3cd8ef46ad1b100e0d3c7e38774f330726820
Author: Ben Hutchings <ben@decadent.org.uk>
Date: Tue Jun 4 19:00:39 2019 +0100

    Revert "consolemap: Fix a memory leaking bug in
drivers/tty/vt/consolemap.c"

    This reverts commit 84ecc2f6eb1cb12e6d44818f94fa49b50f06e6ac.

So NACK.

Do we have some annotations for this instead?

```
> Signed-off-by: James Bond <jameslouisebond@gmail.com>
> ---
>  drivers/tty/vt/consolemap.c | 5 ++++-
>  1 file changed, 4 insertions(+), 1 deletion(-)
>
> diff --git a/drivers/tty/vt/consolemap.c b/drivers/tty/vt/consolemap.c
> index 5947b54d92be..1e8d06c32ca1 100644
> --- a/drivers/tty/vt/consolemap.c
> +++ b/drivers/tty/vt/consolemap.c
> @@ -489,7 +489,10 @@ con_insert_unipair(struct uni_pagedir *p, u_short unicode, u_short
fontpos)
>       p2 = p1[n = (unicode >> 6) & 0x1f];
>       if (!p2) {
>               p2 = p1[n] = kmalloc_array(64, sizeof(u16), GFP_KERNEL);
> -             if (!p2) return -ENOMEM;
> +             if (!p2) {
> +                     kfree(p1);
> +                     return -ENOMEM;
> +             }
>               memset(p2, 0xff, 64*sizeof(u16)); /* No glyphs for the characters (yet) */
>       }
```

thanks,
--
js
suse labs

3) https://lore.kernel.org/lkml/20200810075122.GA1531406@kroah.com/

Greg Kroah-Hartman <gregkh@linuxfoundation.org>
Mon, Aug 10, 2020, 2:51 AM
to Jiri, me, Andrew, Kees, Michel, Vlastimil, Denis, linux-kernel, Ben

I have a whole talk just about the "fun" involved with that change:
https://kernel-recipes.org/en/2019/talks/cves-are-dead-long-live-the-cve/

> Do we have some annotations for this instead?

We need something there, a comment saying "this is fine, don't touch
it!" or something like that? We need that in a few other places in the
vt code as well.

> > Signed-off-by: James Bond <jameslouisebond@gmail.com>

Nice name...

James, can you mark this syzbot report as "invalid" or something like
that please so that this does not come up again and again.

thanks,

greg k-h

4) https://lore.kernel.org/lkml/08df63cd-c4b9-c16b-2e27-5d86580eebf2@kernel.org/
Jiri Slaby jirislaby@kernel.org via gmail.com
Aug 10, 2020, 3:14 AM
to Greg, me, Andrew, Kees, Michel, Vlastimil, Denis, linux-kernel, Ben

...
>> Do we have some annotations for this instead?
>
> We need something there, a comment saying "this is fine, don't touch
> it!" or something like that? We need that in a few other places in the
> vt code as well.

Sure, comment as the last resort (to silence patch writers). But I had
some kmemleak annotation (to silence the warning) in mind.

Or better fix/tune kmemleak: why it dares to think it's a mem leak in
the first place?

thanks,
--
js

5) (this message from Qiushi Wu was dropped by LKML due to it not in plaintext mode)
James Bond <jameslouisebond@gmail.com>

Aug 10, 2020, 12:34 PM
to Ben, Jiri, Greg, Andrew, Kees, Michel, Vlastimil, Denis, linux-kernel

>> We need something there, a comment saying "this is fine, don't touch
>> it!" or something like that? We need that in a few other places in the
>> vt code as well.

> Or better fix/tune kmemleak: why it dares to think it's a mem leak in
> the first place?

It's also possible that the Kmemleak indeed identified a memory leak for the pointer p1,
but the leak point is at another place we don't find.
The current fixed point happens to look like the leak point (but actually not) because
Kmemleak can only identify the allocation of the leaked pointer,
but not the place where it misses free, which further requires manual analysis.

>> James, can you mark this syzbot report as "invalid" or something like
>> that please so that this does not come up again and again.

Sure.

Thanks,
James

Patch 3

1) https://lore.kernel.org/lkml/20200821034458.22472-1-acostag.ubuntu@gmail.com/

From: George Acosta <acostag.ubuntu@gmail.com>
To: acostag.ubuntu@gmail.com
Cc: Matt Porter <mporter@kernel.crashing.org>,
Alexandre Bounine <alex.bou9@gmail.com>,
Andrew Morton <akpm@linux-foundation.org>,
John Hubbard <jhubbard@nvidia.com>,
"Gustavo A. R. Silva" <gustavoars@kernel.org>,
Madhuparna Bhowmik <madhuparnabhowmik10@gmail.com>,
Kees Cook <keescook@chromium.org>,
Dan Carpenter <dan.carpenter@oracle.com>,
linux-kernel@vger.kernel.org
Subject: [PATCH] rapidio: fix get device imbalance on error
Date: Thu, 20 Aug 2020 22:44:57 -0500
Message-ID: <20200821034458.22472-1-acostag.ubuntu@gmail.com> (raw)

Fix the imbalance in mport_cdev_open.

```
Call put_device in error path to balance the
refcount that increased by the get_device.

Fixes: e8de370188d0 ("rapidio: add mport char device driver")
Signed-off-by: George Acosta <acostag.ubuntu@gmail.com>
---
 drivers/rapidio/devices/rio_mport_cdev.c | 1 +
 1 file changed, 1 insertion(+)

diff --git a/drivers/rapidio/devices/rio_mport_cdev.c b/drivers/rapidio/devices/rio_mport_cdev.c
index c07ceec3c6d4..3b68e00eb40f 100644
--- a/drivers/rapidio/devices/rio_mport_cdev.c
+++ b/drivers/rapidio/devices/rio_mport_cdev.c
@@ -1908,6 +1908,7 @@ static int mport_cdev_open(struct inode *inode, struct file *filp)
                                sizeof(struct rio_event) * MPORT_EVENT_DEPTH,
                                GFP_KERNEL);
        if (ret < 0) {
+               put_device(&chdev->dev);
                dev_err(&chdev->dev, DRV_NAME ": kfifo_alloc failed\n");
                ret = -ENOMEM;
                goto err_fifo;
--
2.17.1
```

2) https://lore.kernel.org/lkml/20200821081449.GI5493@kadam/

```
Subject: Re: [PATCH] rapidio: fix get device imbalance on error
Date: Fri, 21 Aug 2020 11:14:49 +0300
Message-ID: <20200821081449.GI5493@kadam> (raw)
In-Reply-To: <20200821034458.22472-1-acostag.ubuntu@gmail.com>

On Thu, Aug 20, 2020 at 10:44:57PM -0500, George Acosta wrote:
> Fix the imbalance in mport_cdev_open.
> Call put_device in error path to balance the
> refcount that increased by the get_device.
>
> Fixes: e8de370188d0 ("rapidio: add mport char device driver")
> Signed-off-by: George Acosta <acostag.ubuntu@gmail.com>
> ---
>  drivers/rapidio/devices/rio_mport_cdev.c | 1 +
>  1 file changed, 1 insertion(+)
>
> diff --git a/drivers/rapidio/devices/rio_mport_cdev.c b/drivers/rapidio/devices/rio_mport_cdev.c
> index c07ceec3c6d4..3b68e00eb40f 100644
> --- a/drivers/rapidio/devices/rio_mport_cdev.c
```

```
> +++ b/drivers/rapidio/devices/rio_mport_cdev.c
> @@ -1908,6 +1908,7 @@ static int mport_cdev_open(struct inode *inode, struct file *filp)
>                                 sizeof(struct rio_event) * MPORT_EVENT_DEPTH,
>                                 GFP_KERNEL);
>         if (ret < 0) {
> +               put_device(&chdev->dev);
>                 dev_err(&chdev->dev, DRV_NAME ": kfifo_alloc failed\n");
>                 ret = -ENOMEM;
>                 goto err_fifo;
> --
```

If we ever hit this error path then the:

```
        list_add_tail(&priv->list, &chdev->file_list);
```

needs to be removed from the list as well or it lead to a use after free. Probably just move the list_add_tail() after the kfifo_alloc() has succeeded. We need to remove it from the list in mport_cdev_release() as well...

The error handling in this function is kind of rubbish.
1) Get rid of the out label and return directly (more readable).
1b) Use "return 0;" instead of "retur ret;" for the success path.
2) Name the labels after what they do not where they "come from". In other words do.

```
err_priv:
        kfree(priv);
```

3) Create a label to call put_device:

```
err_device:
        put_device(&chdev->dev);
```

Change all the error paths to use the goto instead of calling put_device() before the goto.

regards,
dan carpenter

Patch 4

1) https://lkml.org/lkml/2020/8/21/124

James Bond <jameslouisebond@gmail.com>
Aug 21, 2020, 2:05 AM

to me, Miquel, Richard, Vignesh, Arnd, Ryan, David, linux-mtd, linux-kernel

pci_dev_get increases the refcount of "pdev".
In the error paths, pci_dev_put should be called
to handle the "pdev" and decrease the corresponding refcount.

Fixes: 90afffc8bd79 ("[MTD] [MAPS] Support for BIOS flash chips on the nvidia ck804 southbridge")
Signed-off-by: James Bond <jameslouisebond@gmail.com>
---
 drivers/mtd/maps/ck804xrom.c | 3 +++
 1 file changed, 3 insertions(+)

```
diff --git a/drivers/mtd/maps/ck804xrom.c b/drivers/mtd/maps/ck804xrom.c
index 460494212f6a..16af8b5ee653 100644
--- a/drivers/mtd/maps/ck804xrom.c
+++ b/drivers/mtd/maps/ck804xrom.c
@@ -195,6 +195,7 @@ static int __init ck804xrom_init_one(struct pci_dev *pdev,
        if (!window->virt) {
                printk(KERN_ERR MOD_NAME ": ioremap(%08lx, %08lx) failed\n",
                       window->phys, window->size);
+               pci_dev_put(pdev);
                goto out;
        }

@@ -222,6 +223,7 @@ static int __init ck804xrom_init_one(struct pci_dev *pdev,

                if (!map) {
                        printk(KERN_ERR MOD_NAME ": kmalloc failed");
+                       pci_dev_put(pdev);
                        goto out;
                }
                memset(map, 0, sizeof(*map));
@@ -295,6 +297,7 @@ static int __init ck804xrom_init_one(struct pci_dev *pdev,
                if (mtd_device_register(map->mtd, NULL, 0)) {
                        map_destroy(map->mtd);
                        map->mtd = NULL;
+                       pci_dev_put(pdev);
                        goto out;
                }

--
```

2) https://lkml.org/lkml/2020/8/27/614

Miquel Raynal <miquel.raynal@bootlin.com>
Aug 27, 2020, 7:46 AM
to Vignesh, Arnd, Ryan, Richard, linux-kernel, linux-mtd, David, me

Hi James,

James Bond <jameslouisebond@gmail.com> wrote on Fri, 21 Aug 2020 02:05:36 -0500:
I suppose in these three cases, the window->maps list will be empty and you will end up returning -ENODEV and the bottom of the function? If yes, it woudl probably be better to move these pci_dev_put() calls to this location.

Otherwise, it might bit interesting to clean up a little bit the error path and perhaps have a distinct success vs. failure path.

Thanks,
Miquèl

3) https://lkml.org/lkml/2020/8/28/1046

James Bond <jameslouisebond@gmail.com>
Aug 28, 2020, 9:28 PM
to Miquel, Vignesh, Arnd, Ryan, Richard, linux-kernel, linux-mtd, David

Hi Miquèl,
Thanks for your feedback.
I have just rechecked this function and find that "pdev" currently is already put
inside ck804xrom_cleanup, so my previous patch is meaningless...
The current calling order is like:

```
window->pdev = pci_dev_get(pdev);
...
    ck804xrom_cleanup(window)
            ->
                pci_dev_put(window->pdev);
```

And sorry for bothering you.
Thanks,
James

4) https://lkml.org/lkml/2020/8/31/362

Hi James,

James Bond <jameslouisebond@gmail.com> wrote on Fri, 28 Aug 2020 21:28:40 -0500:

> Hi Miquèl,
> Thanks for your feedback.
> I have just rechecked this function and find that "pdev" currently is
> already put
> inside ck804xrom_cleanup, so my previous patch is meaningless...
> The current calling order is like:
>
> window->pdev = pci_dev_get(pdev);
> ...
> ck804xrom_cleanup(window)
> ->
> pci_dev_put(window->pdev);
> And sorry for bothering you.

Ok that's fine, thanks anyway!

Miquèl

Patch 5

https://lore.kernel.org/lkml/20200804183650.4024-1-jameslouisebond@gmail.com/
From: James Bond <jameslouisebond@gmail.com>
To: jameslouisebond@gmail.com
Cc: "Gustavo A. R. Silva" <gustavo@embeddedor.com>,
 Greg Kroah-Hartman <gregkh@suse.de>,
 Mike Waychison <mikew@google.com>,
 linux-kernel@vger.kernel.org
Subject: [PATCH] firmware: dmi-sysfs: Add clean-up operations to fix refcount leak
Date: Tue, 4 Aug 2020 13:36:49 -0500
Message-ID: <20200804183650.4024-1-jameslouisebond@gmail.com> (raw)

According to the documentation of function kobject_init_and_add(),
when this function returns an error, kobject_put() must be called
to properly clean up the memory associated with the object.

Fixes: 925a1da7477f ("firmware: Break out system_event_log in dmi-sysfs")
Signed-off-by: James Bond <jameslouisebond@gmail.com>
---
 drivers/firmware/dmi-sysfs.c | 4 +++-

```
 1 file changed, 3 insertions(+), 1 deletion(-)

diff --git a/drivers/firmware/dmi-sysfs.c b/drivers/firmware/dmi-sysfs.c
index 8b8127fa8955..848b6a0f94eb 100644
--- a/drivers/firmware/dmi-sysfs.c
+++ b/drivers/firmware/dmi-sysfs.c
@@ -457,8 +457,10 @@ static int dmi_system_event_log(struct dmi_sysfs_entry *entry)
                                        &dmi_system_event_log_ktype,
                                        &entry->kobj,
                                        "system_event_log");
-       if (ret)
+       if (ret) {
+               kobject_put(entry->child);
                goto out_free;
+       }

        ret = sysfs_create_bin_file(entry->child, &dmi_sel_raw_attr);
        if (ret)
--
2.17.1
```

End