tianocore

Reporting Issues

Bug 1743 (CVE-2021-28210) - Unlimited FV Recursion, round 2

Status: RESOLVED FIXED

Alias: CVE-2021-28210

Product: Tianocore Security Issues
Component: Security Issue (show other bugs)
Version: Current
Hardware: All All

Importance: Lowest normal
Assignee: Laszlo Ersek

URL:
Keywords:

Depends on:
Blocks:

Reported: 2019-04-23 13:04 UTC by Laszlo Ersek
Modified: 2021-03-12 16:53 UTC (History)
CC List: 12 users (show)

See Also: ~~CVE-2018-12183~~
Release(s) the issue is observed: edk2-stable202008
The OS the target platform is running: ---
Package: MdeModulePkg
Release(s) the issues must be fixed: edk2-stable202011

Attachments	
TestCase on OVMF (3.50 KB, text/plain) 2020-09-28 22:19 UTC, gaoliming	Details
[PATCH v2 0/2] MdeModulePkg/Core/Dxe: limit FwVol encapsulation section recursion (5.54 KB, application/zip) 2020-09-29 14:44 UTC, Laszlo Ersek	Details
CVE .json file_2 (911 bytes, application/json) 2021-03-04 13:41 UTC, kevinj	Details
Add an attachment (proposed patch, testcase, etc.)	Show Obsolete (2)

Note
You need to log in before you can comment on or make changes to this bug.

Laszlo Ersek 2019-04-23 13:04:25 UTC Description

In <~~https://bugzilla.tianocore.org/show_bug.cgi?id=1174#c6~~>, I asked two questions about the unlimited FV parsing recursion in the PEI Core. That was two weeks ago and I haven't received an answer.

In summary, I made two points:

(1) The PEI Core issue was real, but it was limited, in practice, to platforms that allowed firmware updates from the "inside". IOW, if a platform doesn't allow a platform user to expose the PEI Core to arbitrary Firmware Volume Descriptor HOBs, then the platform isn't vulnerable (the unbounded recursion in the PEI Core cannot be triggered by firmware volumes crafted by the attacker).

(2) Even on platforms where the PEI Core is indeed exposed, the PEI Stack Guard feature (~~bug 1126~~) is an unfit solution to the problem. The recursion depth in the PEI phase parsing should be tracked explicitly, and limited by a platform-specific constant (PCD).

I didn't receive a confirmation for (1), and I received no opinions on (2).

Now, in addition to the above (still open, but public) questions, I have another (not public, ATM) question too. In my opinion, the same unbounded recursion exists in the DXE Core as well, except with more serious exposure:

(a) The DXE Core sets up a protocol notify function in its entry point, for instances of the Firmware Volume Block2 Protocol:

```
DxeMain()            [MdeModulePkg/Core/Dxe/DxeMain/DxeMain.c]
  FwVolDriverInit()  [MdeModulePkg/Core/Dxe/FwVol/FwVol.c]
```

(b) Assume that a 3rd party UEFI driver or application installs an FVB instance, with crafted contents. The notification function runs:

```
NotifyFwVolBlock() [MdeModulePkg/Core/Dxe/FwVol/FwVol.c]
```

installing an instance of the Firmware Volume 2 Protocol on the handle.

(c) The EFI_FIRMWARE_VOLUME2_PROTOCOL.ReadSection() member performs "a depth-first, left-to-right search algorithm through all sections found in the specified file" (quoting the PI spec), as follows:

```
FvReadFileSection()   [MdeModulePkg/Core/Dxe/FwVol/FwVolRead.c]
  GetSection()
[MdeModulePkg/Core/Dxe/SectionExtraction/CoreSectionExtraction.c]
    FindChildNode()
[MdeModulePkg/Core/Dxe/SectionExtraction/CoreSectionExtraction.c]
      FindChildNode() // recursive call
```

FindChildNode() is called recursively for encapsulation sections.

Therefore my new question is:

(3) Can you please confirm that the above issue is a vulnerability that is similar in impact to (or worse than) ~~bug 1177~~?

If so, I suggest that we fix it with the "explicit recursion limit" approach that I suggested under (2).

(In the PI spec v1.7, Vol 3, section "2.1.5 Firmware File Sections" says,

"The file image itself can be thought of as the root and may contain an arbitrary number of sections".

That is, the depth is unlimited per spec, so in edk2 a non-recursive (=iterative) traversal would be safer. Still, for a simpler fix, an explicit maximum depth should be OK. EFI_OUT_OF_RESOURCES would be a suitable & compliant return value.)

Unlike ~~bug 1177~~, the present report applies to ArmVirtPkg and OvmfPkg platforms.

Thanks!

Vincent Zimmer 2019-05-29 12:55:11 UTC Comment 1

5/29/19 tiano infosec scrub - jian to look at this. synch w/ owner who addressed pei variant

Bret Barkelew 2019-06-25 16:45:54 UTC Comment 2

If an iterative traversal would be safer, would it be better to aim for that rather than an arbitrary limit?

Also, is there merit in just shutting down the notification prior to 3rd party code, or do we expect that an intended behavior is that 3rd party code can publish FVs. I would argue that that's a corner case, at best.

Laszlo Ersek 2019-06-26 05:45:41 UTC Comment 3

(In reply to Bret Barkelew from comment #2)
> If an iterative traversal would be safer, would it be better to aim for that
> rather than an arbitrary limit?

This is up to the module owner(s) -- personally I think the iterative traversal would bring too little benefit in comparison to the complexity of the patch(es).

Maybe a far-fetched example, but, on a POSIX system, open() can fail with -1/ELOOP if a symbolic link loop is encountered during pathname resolution. However, an actual loop may not be determined in the chain, it may be enough if the system *assumes* a loop, based on the number of symbolic links seen (SYMLOOP_MAX).

http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/limits.h.html
http://pubs.opengroup.org/onlinepubs/9699919799/functions/openat.html

> Also, is there merit in just shutting down the notification prior to 3rd
> party code, or do we expect that an intended behavior is that 3rd party code
> can publish FVs. I would argue that that's a corner case, at best.

I think I agree. It seems reasonable to disable the FVB protocol notify at End-of-Dxe.

(Currently, FwVolDriverInit() calls EfiCreateProtocolNotifyEvent() to establish the FVB protocol notify. EfiCreateProtocolNotifyEvent() does not output the event created for this, so other parts of the code couldn't call CloseEvent() on that event, at the moment, in order to shut down the notify.)

Laszlo Ersek 2019-08-09 09:44:07 UTC Comment 4

Setting need_info on Jian, per comment 1.

Bret Barkelew 2020-04-01 12:35:57 UTC Comment 5

Laszlo (sorry for the ~9 month delay in response),

I'm good with your explanation of the recursion limit.

If we're both happy with shutting down the notification, should we have two patches: a primary that shuts down the notification at EoD and a second that places a limit (as a safety net in case some platform elects to preserve the notification)?

Laszlo Ersek 2020-04-01 17:01:05 UTC Comment 6

(In reply to Bret Barkelew from comment #5)

> If we're both happy with shutting down the notification, should we have two
> patches: a primary that shuts down the notification at EoD and a second that
> places a limit (as a safety net in case some platform elects to preserve the
> notification)?

Sounds good to me, thank you!

Vincent Zimmer 2020-05-06 11:18:38 UTC Comment 7

discussed in 4/1/20 infosec meeting

need more info

no patch

leave in 'unconfirmed'

dxe verification logic uses existence of the FVB to make some policy decisions.

As such, ability to publish is an important security boundary.
This would benefit from some renewed attention.

Next step is to poke the maintainers (start w/ Liming) on this area to read in on the issue.

Least-privilege approach of not being able to publish the fvb post end-of-dxe is a good clean-up.

Vincent Zimmer 2020-05-06 11:38:29 UTC Comment 8

await liming feedback

Laszlo Ersek 2020-09-25 11:01:16 UTC Comment 9

Liming -- can you please comment?

Laszlo Ersek 2020-09-25 11:03:21 UTC Comment 10

Also I think we'll need a new CVE number for this.

gaoliming 2020-09-27 00:42:44 UTC Comment 11

PI spec vol 2 Section 7.3 Dispatcher Services provides DXE service: Dispatch() and ProcessFirmwareVolumn(). They can be used after end of DXE. If the notification even shutdonw on end of DXE, gDS->ProcessFirmwareVolumn() will not work any more. And, per BZ 1126, stack guard is ready in UEFI/SMM. So, I think this issue can also be handled by stack guard.

Laszlo Ersek 2020-09-28 08:58:04 UTC Comment 12

Hi Liming,

PcdCpuStackGuard defaults to FALSE, and the relevant commit message suggests enabling it only conditionally:

commit 448d014b7359b92404653a4da3c63abe4d2389a5
Author: Jian J Wang <jian.j.wang@intel.com>
Date: Thu Oct 12 12:28:47 2017 +0800

    MdeModulePkg/metafile: Add PCD PcdCpuStackGuard

    PcdCpuStackGuard is introduced to enable/disable Stack Guard feature.
    Its value is FALSE by default. This feature is suggested to be enabled
    only if the cpu driver and CpuExceptionHandlerLib have supported stack

switch for the processor used in platform. Otherwise the exception dump
message won't be printed out when there's a stack overflow happened.

Furthermore, even if the stack guard were technically capable of catching this
issue, it's still not graceful behavior.

Laszlo Ersek 2020-09-28 11:38:22 UTC Comment 13

Created attachment 563 [details]
[PATCH 0/2] MdeModulePkg/Core/Dxe: limit FwVol encapsulation section recursion

Proposed patch set. Applies on top of commit 1d058c3e86b0 ("IntelFsp2Pkg
GenCfgOpt.py: Initialize IncLines as empty list", 2020-09-25).

Laszlo Ersek 2020-09-28 11:41:49 UTC Comment 14

(In reply to Laszlo Ersek from comment #13)
> Created attachment 563 [details]
> [PATCH 0/2] MdeModulePkg/Core/Dxe: limit FwVol encapsulation section
> recursion
>
> Proposed patch set. Applies on top of commit 1d058c3e86b0 ("IntelFsp2Pkg
> GenCfgOpt.py: Initialize IncLines as empty list", 2020-09-25).

Cc: Dandan Bi <dandan.bi@intel.com>
Cc: Hao A Wu <hao.a.wu@intel.com>
Cc: Jian J Wang <jian.j.wang@intel.com>
Cc: Liming Gao <gaoliming@byosoft.com.cn>
Cc: Philippe Mathieu-Daudé <philmd@redhat.com>

Please review.

Note: I have only build-tested this patch series. I don't even have a reproducer
that leads to the execution of FvReadFileSection()
[MdeModulePkg/Core/Dxe/FwVol/FwVolRead.c].

If someone can help with an FDF file and/or maybe a UEFI application that leads to
the traversal of nested sections, that would be appreciated too.

Thanks.

gaoliming 2020-09-28 22:19:49 UTC Comment 15

Created attachment 565 [details]
TestCase on OVMF

gaoliming 2020-09-28 22:22:44 UTC Comment 16

Attach the test case on OVMF platform. It can be used to verify this patch.

For this patch, I suggest to update PcdFwVolDxeMaxEncapsulationDepth default value
to a bigger value, such as 0x10. The real case may meet with the depth of 8.

Laszlo Ersek 2020-09-29 12:54:32 UTC Comment 17

Awesome; thank you, Liming for the tester!

Laszlo Ersek 2020-09-29 13:53:01 UTC Comment 18

After implementing the update Liming is suggesting in comment 16 (= DEC default of
the PCD should be 0x10), the reproducer from comment 15 reports success:

```
>| FS0:\> HelloWorld.efi
>| UEFI Hello World!
>| FS0:\> echo %lasterror%
>| 0x0
```

In order to test the limit, rebuild Ovmf IA32X64 with the following flag:

--pcd gEfiMdeModulePkgTokenSpaceGuid.PcdFwVolDxeMaxEncapsulationDepth=8

(note that the test application need not be rebuilt). Then the output is:

```
>| FS0:\> HelloWorld.efi
>| UEFI Hello World!
>| FS0:\> echo %lasterror%
>| 0xE
```

where 0xE stands for EFI_NOT_FOUND from GetSectionFromAnyFv(). Additionally, the
firmware log contains

```
>| GetSection: recursion aborted due to nesting depth
```

Laszlo Ersek 2020-09-29 14:44:53 UTC Comment 19

Created attachment 566 [details]
[PATCH v2 0/2] MdeModulePkg/Core/Dxe: limit FwVol encapsulation section recursion

Proposed v2 patch set, addressing Liming's feedback from comment 16.

See the v1->v2 changes in the patch files (in the Notes sections).

Applies on top of commit 52dbaaeace64 ("CryptoPkg/BaseCryptLib: add crypto
algorithms needed by variable protection", 2020-09-29).

gaoliming 2020-09-29 20:41:43 UTC Comment 20

This version is good to me. Reviewed-by: Liming Gao <gaoliming@byosoft.com.cn>

Laszlo Ersek 2020-10-01 03:42:33 UTC Comment 21

Thank you, Liming! I'm going to ping edk2-infosec about the next steps.

Philippe Mathieu-Daudé 2020-10-01 07:15:53 UTC Comment 22

(In reply to Laszlo Ersek from comment #21)

In patch #1:

```
   @param  SectionInstance        Indicates which instance of section to find.
                                  This is an in/out parameter and it is 1-based,
                                  to deal with recursions.
```

it was not easy to me to understand the "it is 1-based".
Suggestion for something clearer (but probably English incorrect):

```
                                  This is an in/out parameter to deal with
                                  recursions, which depth base is 1.
```

Anyway the rest is OK, thanks for fixing this issue.

To both patches:
Reviewed-by: Philippe Mathieu-Daude <philmd@redhat.com>

Laszlo Ersek 2020-10-02 03:40:15 UTC Comment 23

Hi Phil,

thanks for the review.

I feel that "1-based" is pretty clear; it means that the depth is given after being added to 1 as a basis. So if you are looking for a particular SectionInstance per EFI_FIRMWARE_VOLUME2_PROTOCOL.ReadSection() interface contract, which is zero-based, then you need to rebase the section index to 1 -- that is, add 1.

This is quite standard language in both the PI spec, and the edk2 codebase.

From the PI v1.7 specification, regarding EFI_FIRMWARE_VOLUME2_PROTOCOL.ReadSection():

>| SectionInstance
>|
>| Indicates which instance of sections with a type of SectionType
>| to return. SectionType in conjunction with SectionInstance
>| indicates which section to return. SectionInstance is zero based.

The same is repeated in the corresponding header file "MdePkg/Include/Protocol/FirmwareVolume2.h":

>| @param SectionInstance   Indicates which instance of sections
>| with a type of SectionType to return.
>| SectionType in conjunction with
>| SectionInstance indicates which
>| section to return. SectionInstance is
>| zero based.

Both the "0-based" and "1-based" expressions are used multiple times in edk2:

>| $ git grep -c '0-based'
>| MdeModulePkg/Bus/Pci/NvmExpressDxe/NvmExpress.h:6
>| MdeModulePkg/Bus/Pci/NvmExpressPei/NvmExpressPei.h:4
>| MdeModulePkg/Bus/Pci/NvmExpressPei/NvmExpressPeiHci.c:1
>| MdeModulePkg/Bus/Pci/NvmExpressPei/NvmExpressPeiHci.h:1
>| MdePkg/Include/Library/UefiLib.h:1
>| MdePkg/Include/Protocol/MmSwDispatch.h:1
>| MdePkg/Include/Protocol/SmmSwDispatch2.h:1
>| MdePkg/Library/UefiLib/UefiNotTiano.c:1
>| NetworkPkg/HttpBootDxe/HttpBootDxe.h:2
>| NetworkPkg/UefiPxeBcDxe/PxeBcImpl.h:2
>| OvmfPkg/Csm/Include/Protocol/LegacyBios.h:1
>| OvmfPkg/Csm/LegacyBiosDxe/LegacyBiosInterface.h:1
>| OvmfPkg/Csm/LegacyBiosDxe/LegacyBootSupport.c:1
>| OvmfPkg/Library/QemuBootOrderLib/QemuBootOrderLib.c:1
>| SecurityPkg/HddPassword/HddPasswordDxe.c:1

(25 total)

>| $ git grep -c '1-based'
>| BaseTools/Source/C/Include/Common/MdeModuleHii.h:1
>| MdeModulePkg/Bus/Usb/UsbBusDxe/UsbEnumer.c:1
>| MdeModulePkg/Include/Guid/MdeModuleHii.h:1
>| MdeModulePkg/Universal/SetupBrowserDxe/Setup.h:1
>| NetworkPkg/HttpBootDxe/HttpBootDxe.h:1
>| NetworkPkg/UefiPxeBcDxe/PxeBcImpl.h:1
>| OvmfPkg/Csm/Include/Framework/FrameworkInternalFormRepresentation.h:1
>| OvmfPkg/Library/QemuBootOrderLib/QemuBootOrderLib.c:1

(8 total, without the attached patch applied)

I'm OK to replace the comment, but then please propose something that I can take verbatim. Personally I think the current patch is sufficiently clear; if an improvement is preferred, please offer something that needs no further corrections (grammatical or otherwise) on top. Thanks!

Philippe Mathieu-Daudé 2020-10-02 04:33:32 UTC Comment 24

(In reply to Laszlo Ersek from comment #23)
[...]
> This is quite standard language in both the PI spec, and the edk2
> codebase.

[...]
> I'm OK to replace the comment, but then please propose something that I
> can take verbatim. Personally I think the current patch is sufficiently
> clear; if an improvement is preferred, please offer something that needs
> no further corrections (grammatical or otherwise) on top. Thanks!

Thanks for enlightening me. I now agree the current patch is sufficiently clear.
No need to update your patch, thanks.

Laszlo Ersek 2020-10-02 04:51:12 UTC Comment 25

Thank you!

Laszlo Ersek 2020-11-12 12:16:51 UTC Comment 26

PROPOSED PUBLIC DATE (opening up the BZ and posting the patch to edk2-devel):

  Thursday 2020-Nov-19 07:00 UTC

in order to get the fix into edk2-stable202011.

Bret Barkelew 2020-11-12 12:59:03 UTC Comment 27

https://media.giphy.com/media/xT8qB3utUzMWqmpH20/source.gif

Riccardo Schirone 2020-11-13 06:12:33 UTC Comment 28

Is this going to have a CVE before it goes public?

Laszlo Ersek 2020-11-13 15:19:48 UTC Comment 29

(In reply to Riccardo Schirone from comment #28)
> Is this going to have a CVE before it goes public?

CC'ing Eric.

Laszlo Ersek 2020-11-19 05:56:07 UTC Comment 30

(In reply to Laszlo Ersek from comment #19)
> Created attachment 566 [details]
> [PATCH v2 0/2] MdeModulePkg/Core/Dxe: limit FwVol encapsulation section
> recursion
>

> Proposed v2 patch set, addressing Liming's feedback from comment 16.
>
> See the v1->v2 changes in the patch files (in the Notes sections).
>
> Applies on top of commit 52dbaaeace64 ("CryptoPkg/BaseCryptLib: add crypto
> algorithms needed by variable protection", 2020-09-29).

Public posting:

* [edk2-devel] [PATCH v2 RESEND 0/2]
  security fix: unlimited FV recursion, round 2 (DXE Core)

msgid <20201119105340.16225-1-lersek@redhat.com>
https://edk2.groups.io/g/devel/message/67707
https://www.redhat.com/archives/edk2-devel-archive/2020-November/msg00865.html

Laszlo Ersek 2020-11-20 20:43:28 UTC Comment 31

Merged as commit range 6c8dd15c4ae4..47343af30435, via
<https://github.com/tianocore/edk2/pull/1137>.

kevinj 2021-03-03 11:44:53 UTC Comment 32

Created attachment 657 [details]
CVE .json file

I have attached the .json file for CVE classification. Please review and provide
feedback.

Laszlo Ersek 2021-03-03 12:04:33 UTC Comment 33

$ git tag --contains 47343af30435
edk2-stable202011

Thus:

last vulnerable release: edk2-stable202008
first fixed release: edk2-stable202011

Laszlo Ersek 2021-03-03 12:07:39 UTC Comment 34

CWE-674 looks like a precise match for this issue.

kevinj 2021-03-04 13:41:18 UTC Comment 35

Created attachment 665 [details]
CVE .json file_2

Thank you Laszlo for your feedback. I have updated the version in the .json file
and re-uploaded it.

Laszlo Ersek 2021-03-12 16:01:54 UTC Comment 36

Thanks for the CVE number, Kevin!

kevinj 2021-03-12 16:04:04 UTC Comment 37

Laszlo,

Please review the .json file again, especially the version this bug is observed in
and inform me when you plan to publicly disclose this bug, so we know when to
submit this CVE back to MITRE. Thank you!

Laszlo Ersek 2021-03-12 16:53:38 UTC Comment 38

Hi Kevin,

(1) The JSON file names edk2-stable202008 (twice), which is indeed the last
vulnerable release, per my comment 33. So that looks OK.

(2) The issue has been disclosed publicly a while ago, please see comment 26 and
comment 30 -- posting the upstream patch qualifies as public disclosure. It
happened on November 19, 2020. If anything administrative depends on the public
disclosure, it should not be delayed.

Thanks!
Laszlo