tianocore

Security Advisory

- for the open source community

TABLE OF CONTENTS

Security Advisory

Version: .004.0

04/30/2025 09:30:21

This document will list briefings on each security issue found and give a description, a recomendation on a solution, an acknowlegment that the solution is validated and references.

Acknowledgements

Process

(short form)

1. Security Bugs reported through: How to report a Security Issue
2. The issue is evaluated
3. Determine if a Security issue
4. Determine Module
5. Fix and Validate issue
6. If Security, Update Security Advisory (This Document)

Revision History

Revision	Revision History	Date
.002.0	Initial release. Logs 1 - 19	Jan 9, 2015
.003.0	Logs for 20-26	Nov 29, 2016

.003.1	Logs for 21-26 - Fix more for DHCP issue, feedback from Phoenix. - Fix Smm Variable GetInfo function issue, discovered by release test. - Fix GIT hash info, which incorrect stated before. logs 21, 22, 23, 24, 25, 26	Dec 19, 2016
.004.0	Log 27 - Update Update Gitbook Template	Jan 11, 2018

1. INSECURE DEFAULT SECURE BOOT POLICY FOR OPTION ROMS

Description:

In order to help prevent vulnerabilities in secure boot implementations, the default policy for Option ROMs was changed to a more secure value.

Recommendation:

This issue is addressed by EDK2 SVN https://sourceforge.net/p/edk2/code/14607

Acknowledgments:

Reported by the Advanced Threat Research team at Intel Security.

2. INCORRECT PKCS#1V1.5 PADDING VERIFICATION FOR RSA SIGNATURE CHECK

Description:

The implementation of RSA signature verification was vulnerable to a Bleichenbacher RSA signature forgery attack when keys with a small public exponent were used.

Recommendation:

This issue is addressed by EDK2 SVN https://sourceforge.net/p/edk2/code/14309.

Acknowledgments:

Reported by the Advanced Threat Research team at Intel Security.

3. UEFI VARIABLE “REINSTALLATION”

Description:

It had been possible to call the `SetVariable` API at runtime on a variable that did not have `RUNTIME_ACCESS` permission, causing a new variable with the same name/GUID to be created. It was possible for this new variable to be used instead of the original, protected variable.

Recommendation:

This issue is addressed by EDK2 SVN https://sourceforge.net/p/edk2/code/13156.

Acknowledgments:

Reported by the Advanced Threat Research team at Intel Security.

4. OVERWRITE FROM PERFORMANCE DATA VARIABLE

Description:

The variable `PerfDataMemAddr` was used to hold the address of a buffer for performance data, and this variable could be arbitrarily modified by runtime software. This could cause firmware to corrupt its own code/data.

Recommendation:

This is addressed by EDK2 SVN https://sourceforge.net/p/edk2/code/14386.

Acknowledgments:

Reported by the Advanced Threat Research team at Intel Security.

5. COMMBUFFER SMM OVERWRITE/EXPOSURE

Description:

“`CommBuffer`” is the name of a communication mechanism between runtime code and runtime SMM code. Malicious code could set the address of `CommBuffer` such that calls to runtime SMM code would overwrite or expose the contents of SMRAM.

Recommendation:

This issue is addressed by EDK2 SVN https://sourceforge.net/p/edk2/code/13514, 13530, and 14292.

Acknowledgments:

Reported by the Advanced Threat Research team at Intel Security.

6. TOCTOU ISSUE WITH COMMBUFFER

Description:

Values from CommBuffer could be changed by DMA, running in parallel with SMM, leading to a buffer overflow during a memory copy operation.

Recommendation:

This issue is addressed by EDK2 SVN https://sourceforge.net/p/edk2/code/14325 and https://sourceforge.net/p/edk2/code/14379.

Acknowledgments:

Reported by the Advanced Threat Research team at Intel Security.

7. SMRAM OVERWRITE IN FAULT TOLERANT WRITE SMI HANDLER

Description:

The function `SmmFaultTolerantWriteHandler` did not correctly validate inputs. This could result in an overwrite of SMRAM.

Recommendation:

This issue is addressed by EDK2 SVN https://sourceforge.net/p/edk2/code/13518 and https://sourceforge.net/p/edk2/code/13763.

Acknowledgments:

Reported by the Advanced Threat Research team at Intel Security.

8. SMRAM OVERWRITE IN SMMVARIABLEHANDLER

Description:

The function SmmVariableHandler did not correctly validate inputs. This could result in an overwrite of SMRAM.

Recommendation:

This issue is addressed by EDK2 SVN https://sourceforge.net/p/edk2/code/13486 and https://sourceforge.net/p/edk2/code/13534

Acknowledgments:

Reported by the Advanced Threat Research team at Intel Security.

9. INTEGER/HEAP OVERFLOW IN SETVARIABLE

Description:

Incorrect input handling in `VariableServiceSetVariable` could lead to a buffer overflow.

Recommendation:

This issue is addressed by EDK2 SVN https://sourceforge.net/p/edk2/code/14305.

Acknowledgments:

Reported by the Advanced Threat Research team at Intel Security.

10. HEAP OVERFLOW IN UPDATEVARIABLE

Description:

The `UpdateVariable` function performed a copy first and then checked the size. This may be too late.

Recommendation:

This is addressed by EDK2 SVN https://sourceforge.net/p/edk2/code/14323.

Acknowledgments:

Reported by the Advanced Threat Research Team at Intel Security.

11. OVERWRITE FROM FIRMWAREPERFORMANCE VARIABLE

Description:

The `FirmwarePerformance` variable contained an address used to store performance statistics without checking the validity of the target location.

Recommendation:

This is addressed by EDK2 SVN https://sourceforge.net/p/edk2/code/14369.

Acknowledgments:

Reported by the Advanced Threat Research Team at Intel Security.

12. INTEGER/BUFFER OVERFLOW IN TPMDXE DRIVER

Description:

The `MeasureVariable` function calculated the sum of many fields. This could lead to an integer overflow that resulted in a small allocation of memory and a large copy.

Recommendation:

This is addressed by EDK2 SVN https://sourceforge.net/p/edk2/code/14396.

Acknowledgments:

Reported by the Advanced Threat Research Team at Intel Security.

13. PROTECTION OF PHYSICALPRESENCE VARIABLE

Description:

The `PhysicalPresence` variable was used to store commands to the TPM, and commands that should require physical presence could be written to it by software.

Recommendation:

This is addressed in EDK2 SVN https://sourceforge.net/p/edk2/code/14619.

Acknowledgments:

Reported by the Advanced Threat Research Team at Intel Security.

14. BOOT FAILURE RELATED TO UEFI VARIABLE USAGE

Description:

When certain UEFI variables were corrupted, various code would `ASSERT`, preventing successful boot.

Recommendation:

This is addressed by the following EDK2 SVN https://sourceforge.net/p/edk2/code/ commits: 13203, 13297, 13323, 14029, 15330, 15336, 15386, 15407, 15357, 15393, 15401, 15340, 15388, 15338, 15328, 15329, 15334, 15405, 15416, 15360, 15339, 15404, 15426, 15337, 15333, 15385, 15356, 15376, 15385, 15391, 15351, 15544, 15545, 15909, 15910, 15977

Acknowledgments:

Reported by the Advanced Threat Research team at Intel Security and Corey Kallenberg, Xeno Kovah, John Butterworth, and Sam Cornwell of the MITRE Corporation.

References:

• CERT/CC VU#758382

15. BUFFER OVERFLOWS IN CAPSULE UPDATE

Description:

During capsule update processing, a loop will continue adding arbitrarily many values from the capsule `(Fvb->NumBlocks)`. After summation, the final value is multiplied by a static size and used to calculate the size of allocation. This allocation, upon integer overflow, can be small, while the loop that copies data based on values from the capsule will copy a large amount of data. Additionally, the `CapsuleCoalesce` function also contained an integer overflow during summation of the size of the image and descriptor. This also results in a small allocation but a large copy.

Recommendation:

These issues are addressed by SVN https://sourceforge.net/p/edk2/code/15136 and https://sourceforge.net/p/edk2/code/15137.

Acknowledgements:

Reported by Corey Kallenberg, Xeno Kovah, John Butterworth, and Sam Cornwell of the MITRE Corporation.

References:

• CERT/CC VU#552286

16. BOOT FAILURE RELATED TO TPM MEASUREMENTS

Description:

When UEFI Variable storage space is full, the TPM measurement driver could not support making a measurement log and would `ASSERT`, preventing successful boot.

Recommendation:

This is addressed by EDK2 SVN https://sourceforge.net/p/edk2/code/16281.

Acknowledgments:

Reported by Intel

References:

• USRT M1248

17. BUFFER OVERFLOW IN VARIABLE RECLAIM

Description:

The Reclaim function that performs garbage collection on the UEFI variable storage area in flash contained a bug that allowed it to search beyond the bounds of the variable storage area. In this circumstance, a buffer overflow may occur. This may result in elevation of privilege or denial of service.

NOTE: This issue would not normally be exposed. In order to exploit this issue, a separate vulnerability must allow modification of the variable storage area and regions after it, normally stored on SPI flash.

However, because the existing implementation depended upon data outside the variable store, this was considered a security issue and mitigated in code that is intended to be used in production.

Recommendation:

The issue was initially reported in `FSVariable.c`, which is not intended for use in production. EDK2 SVN https://sourceforge.net/p/edk2/code/16217 and https://sourceforge.net/p/edk2/code/16218 introduce comments to clarify that the code is not intended for production use.

Updates to the Reclaim function in `MdeModulePkg` and `SecurityPkg` in EDK2 were made in EDK2 SVN https://sourceforge.net/p/edk2/code/16280.

Acknowledgments:

Reported by Rafal Wojtczuk from Bromium and Corey Kallenberg from MITRE.

References:

• CERT/CC VU#533140 • USRT M1247

18. OVERFLOW IN PROCESSING OF AUTHVARKEYDATABASE

Description:

When the UEFI Variable `AuthVarKeyDatabase` is used, it may be possible to overflow a statically allocated buffer.

Recommendation:

The variable `AuthVarKeyDatabase` is an authenticated variable. Its contents may not be changed without access to an authorized private key. This limits the severity of the issue. EDK2 maintainers have assessed that this issue violates the threat model and does not require explicit mitigation. This issue requires an attacker to introduce inconsistency into internal data structures in the variable storage area. The EDK2 architecture requires internal data structures to be protected in implementation. An attacker capable of bypassing this protection may introduce many other inconsistencies. EDK2 SVN https://sourceforge.net/p/edk2/code/16227 added a comment to the source file in order to clarify this.

Acknowledgments:

Reported by Rafal Wojtczuk from Bromium and Corey Kallenberg from MITRE.

References:

• USRT M1246

19. COUNTER BASED AUTHENTICATED VARIABLE ISSUE

Description:

The variable `AuthVarKeyDatabase` is internally used by the UEFI variable driver and was protected with a counter-based authentication attribute. The implementation used an incorrect public key to process updates to this variable.

Recommendation:

This issue has been addressed by EDK2 SVN https://sourceforge.net/p/edk2/code/16220.

Credit:

Reported by Intel.

References:

• USRT M1290

20. HONORING MEMORY ONLY RESET CONTROL AND CORRECT MOR SPEC IMPLEMENTATION

Disclosure Time line

- July 2015 – Initial notification to affected parties (this document)
- Mid-September 2015 - Patches committed to open source, as needed
- January 2016 – Public advisories added to open source projects, as needed

NOTE: Disclosure plan has been updated in revision 1.2 to allow 6 months before public disclosure, as requested. Product and open source updates should nonetheless strive to address the issue more rapidly, if possible.

This disclosure plan is designed to enable affected parties sufficient time to deploy mitigation for these issues prior to any planned public disclosure.

Background

In EDK II source code, some infrastructure exists for the Memory Overwrite Request (MOR) capability, which allows an OS to request that memory be cleared upon reboot (see the TCG Platform Reset Attack Mitigation Specification Version 1.00, Revision .92 or later). Revision 1.00 is strongly encouraged, including implementation of detecting an orderly OS shutdown to prevent unnecessary memory clear operations. This is a standard published by the Trusted Computing Group and required by some operating systems, such as Microsoft Windows.

The EDK II infrastructure for this capability is implemented in the `SecurityPkg`, as described in page 15 of A Tour Beyond BIOS: Implementing TPM2 Support in EDKII. This generic code can signal to the platform that there must be a ‘clear’ operation, but the underlying platform implementation of the memory clear is often in closed source platform and SI-vendor specific code. In the case of EDK II, the signal is implemented with a UEFI variable. EDK II provides code to initialize, set, and reset the variable, but there also needs to be platform-specific code to honor the signal and perform the clear operation.

Issue -1

We have discovered that, on some platforms, the signal to clear memory is not honored. This could be because the platform-specific code to do the clear is not implemented or because the firmware fails to invoke this code correctly. If the MOR signal is not honored, Cold Boot attacks may be able to more easily reveal encryption keys or other secrets from OS runtime.

Recommendation -1

Firmware developers who base code on EDK II should review their implementation for this issue and correct it, if needed. Where changes to open source are needed, Intel is recommending an embargo according to the timeline above.

Here is some sample code for an EDKII-style PEI Platform code that can do the clear:

```
Status = VariableServices->PeiGetVariable (
          PeiServices,
          EfiMemoryOverwriteControlVariable,
          &gEfiMemoryOverwriteControlDataGuid,
          NULL,
```

```
          &VarSize,
          &MemoryOverwriteReq
          );
if (!EFI_ERROR(Status)) {
// Set a Bit to tell Memory reference code to zeroize memory w/ hardware engine
// or perform the zeroize operation in software
}
```

The actual clear operation will depend upon the platform type, so the above example only has a placeholder within the non-error path to insert the appropriate codes.

Implementation on Quark http://comments.gmane.org/gmane.comp.bios.edk2.devel/7022

> **NOTE TO PUBLICATIONS:** Need to decide if we should await MinnowBoard Max applying HSD 216056 this fix (Fixed in Release 91 of MinnowBoard MAX) now that the 1/16 embargo from USRT has passed. Also, maybe replace above code sample w/ the Galileo impl?

Issue – 2

Document

> **TCG Platform Rest Attack Mitigation Specification**
> Secification Version 1.00
> Revision 1.00
> May 15, 2008
> Pulished

Reads on having the implementation do the following:

> **SetVariable may be modified to disallow deletion of this UEFI variable, and to disallow a change in its attributes.**

this was not the case in the MOR driver for the last several years.

Repro steps:

1. Set BIOS /BIOS/Intel advanced menu/TPM configuration/Current TPM device = dTPM 2.0
2. Set BIOS /BIOS/Boot maintenance menu/Secure boot configuration/Attempt Secure boot = checked
3. Restart the system
4. Run the command 'setvar MemoryOverwriteRequestControl -guid E20939BE-32D4-41BE-A150-897F85D49829 -nv -bs -rt = 01' in EFI_shell
5. Run the command 'dmpstore -b -d -guid E20939BE-32D4-41BE-A150-897F85D49829' in EFI_shell
6. result should could be a failure

Recommendation – 2

This gap was addressed along w/ the implementation of the new feature described in https://msdn.microsoft.com/en-us/library/windows/hardware/mt270973(v=vs.85).aspx in the EDKII implementation

MORLOCK2 checked in at EDK2 SVN https://sourceforge.net/p/edk2/code/19690

Recommend implementations pick up the above patch.

Acknowledgments

This first issue was initially reported by Jeremiah Cox of Microsoft Corporation. Second issue was reported both by Intel internal testing and Jeremiah Cox of Microsoft Corporation.

References

USRT Advisory M1412: Memory-Overwrite-Request (MOR) support

Secure MOR Implementation https://msdn.microsoft.com/en-us/library/windows/hardware/mt270973(v=vs.85).aspx

The TCG Platform Reset Attack Mitigation Specification Version 1.00, Revision .92 or later. Revision 1.00 (http://www.trustedcomputinggroup.org/resources/pc_client_work_group_platform_reset_attack_mitigation_specification_version_10).

Windows 8 Logo requirements http://download.microsoft.com/download/A/D/F/ADF5BEDE-C0FB-4CC0-A3E1- B38093F50BA1/windows8-hardware-cert-requirements-system.pdf.

A Tour Beyond BIOS with the UEFI TPM2 Support in EDKII https://firmware.intel.com/sites/default/files/resources/A_Tour_Beyond_BIOS_Implementing_TPM2_Support_in_EDKII.pdf

Cold Boot Attacks https://citp.princeton.edu/research/memory/

21. TCG PP S4 ISSUE

Description:

TCG physical presence will record TCG PP flag (if user confirmation is required) to a variable `(TCG2_PHYSICAL_PRESENCE_FLAGS_VARIABLE/gEfiTcg2PhysicalPresenceGuid)`. This variable is locked by `EDKII_VARIABLE_LOCK_PROTOCOL`.
In S4 resume path, the code `Tcg2PhysicalPresenceLibProcessRequest` finds it is S4 resume and return immediately. It does not call `EDKII_VARIABLE_LOCK_PROTOCOL` to lock the variable `(TCG2_PHYSICAL_PRESENCE_FLAGS_VARIABLE/gEfiTcg2PhysicalPresenceGuid)`.

Recommendation:

We need update `Tcg2PhysicalPresenceLibProcessRequest` to move S4 check AFTER variable lock.
This is addressed by EDK2 GIT 7b9b576c71c71ed134f50497fd58f862109dd80b.

Acknowledgments:

Reported by Coleman, Rusty Rusty.Coleman@intel.com.

References:

• USRT M1615

22. BIOS PASSWORD

Description:

BIOS setup driver may provide capability for admin password or user password. In Edk II sample driver - `MdeModulePkg\Universal\DriverSampleDxe` , the password is saved to variable. However, this code in this sample driver might be copied to production code.
The `EncodePassword` function only uses a simple XOR with constant key to encode password and save to variable. The variable can be read by anyone. The malicious code to get the variable, and use XOR with this constant key to get the password easily.

Recommendation:

The bad example in EDKII is deleted by GIT 6bfd7ea7d65af28910779b9c72ff2e5fd3a2a54e, 88f0c4e29c03600f2a45a5bd14c500049d2b09dc ..87f04621ad4069c3b2994bc217971d1c5a53fa82.

The better way to encode password is:

1. Generate a random salt value for user.
2. Use SHA256 to hash the password and salt value.
3. Save random salt and hash to variable.

Acknowledgments:

Reported by Matrosov, Alexander alexander.matrosov@intel.com.

References:

• USRT M1617, M1633

23. OPAL DRIVER HAS PP ISSUE ON BLOCKSID

Description:

EDKII open source has OPAL driver at `SecurityPkg\Tcg\Opal`. It includes a feature named `BlockSid`, which is defined in **TCG Physical Presence and TCG OPAL BlockSid specification**. The TCG PP spec defines PP opcode to enable/disable `BlockSid`, which may need user confirmation. However, current EDKII OPAL driver just uses a normal variable `(OPAL_EXTRA_INFO_VAR_NAME/gOpalExtraInfoVariableGuid)` to store the BlockSid enable/disable. This driver does not follow TCG recommendation to use PP process to request user confirmation on BlockSid state change. Also this variable is NOT locked. It means any one can overwrite this variable and bypass BlockSid operation.

Recommendation:

We had better follow TCG PP specification, and implement TCG storage operation (96~101). So that:

1. User confirmation is needed when BlockSid state is changed. (This follows TCG PP spec)
2. This BlockSid state is still saved into variable, but the variable is locked via `EDKII_VARIABLE_LOCK_PROTOCOL` (This makes sure no malicious code may modify BlockSid enable/disable state.)

This is addressed by EDK2 GIT e92ddda2b547f0b952935abaf44fd72e97dbf755..4e3b05a49f454bc257252ae9090421e3c8447737, bee13c00218f3ed3118d8d87683c11b31ca04564, 01dd077315c6759c94af9af4232f8318db13cf8d.

Acknowledgments:

Reported by Yao, Jiewen jiewen.yao@intel.com.

References:

- USRT M1620
- TCG PC Client Platform Physical Presence Interface Specification
- TCG Storage Feature Set: Block SID Authentication

24. OPAL DRIVER HAS PSID ISSUE

Description:

EDKII open source has OPAL driver at `SecurityPkg\Tcg\Opal`. It includes a feature named PSID. PSID is a hard driver specific key which is used to revert to factory default mode. This PSID value should be kept as secret. Current EDKII OPAL driver does not clear PSID in memory after use. The secret value is left in stack, or global variable memory without clear. Technically, a malicious program may search memory and find out the PSID in memory, if user inputs the PSID value in BIOS.

Recommendation:

1. To fix this specific issue, OPAL driver should call `ZeroMem()` to clear PSID in memory after it is used.
2. As generic rule, if a password or other secret is used in a driver, this driver should call `ZeroMem()` to clear secret in memory after it is used.

This is addressed by EDK2 GIT e92ddda2b547f0b952935abaf44fd72e97dbf755..4e3b05a49f454bc257252ae9090421e3c8447737, bee13c00218f3ed3118d8d87683c11b31ca04564, 01dd077315c6759c94af9af4232f8318db13cf8d.

Acknowledgments:

Reported by Yao, Jiewen jiewen.yao@intel.com.

References:

• USRT M1619

25. DHCP MISSES BOUNDARY CHECK FOR NETWORK PACKET

Description:

The UEFI DHCP Protocol has many conventions for processing and caching incoming DHCP4/DHCP6 packets. Their current exists a check in `PxeBcCacheDhcp4Packet` before calling `CopyMem()` on two `EFI_DHCP4_PACKET` structs. This check uses an `ASSERT` which will be compiled out for RELEASE builds of UEFI on EDK II.

But actually, the source is from an external network, and there is no guarantee that the source Length is smaller than destination size. It might happen.

Recommendation:

1. For this specific issue, we need remove `ASSERT` and use error checking.
2. clarify the rule, `ASSERT` can only be used for something **never** happen. Error check must be used for something **might** happen.

This is addressed by EDK2 GIT 4f6b33b460226bc1a54d8af2c0f4fe195f2f04ce, 632dcfd6857b6211ce3fe9755d3c11e74ef5d4477, 471342bbefaac1c21fe7fa4e80949b552b12fbdd, a35dc6499beb0b76c340379a06dff74a8d38095a.

Acknowledgements:

Reported by Timzen, Topher topher.timzen@intel.com

References:

• USRT M1622

26. SMMCORE COMM BUFFER CHECK HAS TOCTOU ISSUE

Description:

A SMM communication buffer is used for data exchange between the SMM component and the non-SMM component. The malicious non-SMM component may generate a malicious SMM communication buffer to attack the SMM component. In order to resist this attack, the SMM component need validate the SMM communication buffer before use it. The validation need cover if the SMM communication buffer overlaps with the SMRAM. The data to be validated should be copied into SMRAM as local variable to avoid Time-Of-Check/Time-Of-Use attack. If the data is not copied into SMRAM, the validation can be bypassed.

For example, a malicious software may let an Application Processor (AP) be outside SMRAM, and only let a Bootstrap Processor (BSP) be inside SMRAM. After the BSP performs the check for the communication buffer, the AP modified the communication buffer. Then when the BSP uses the communication buffer later, the BSP is attacked and the check is actually bypassed.

See `MdeModulePkg/Core/PiSmmCore/PiSmmCore.c` : The function `InternalIsBufferOverlapped` and `SmmIsBufferOutsideSmmValid` are the checker. They validate `gSmmCorePrivate->CommunicationBuffer` and `gSmmCorePrivate->BufferSize`

However these data are outside SMRAM, the malicious code may modify them after check. So below code may still get wrong `gSmmCorePrivate->CommunicationBuffer` .

Recommendation:

We need copy `gSmmCorePrivate->CommunicationBuffer` and `gSmmCorePrivate->BufferSize` to be a local variable, then check and use them, finally sync local variable back to outside SMRAM.

Because the `BufferSize` is moved to SMRAM by SmmCore, the SMM driver should not check the address for the BufferSize.

This is addressed by EDK2 GIT eaae7b33b1cf6b9f21db1636f219c2b6a8d88afd, 62016c1e898434a0326f658912b1e7e0a9c5575e.

Acknowledgments:

Reported by Yao, Jiewen jiewen.yao@intel.com

References:

• USRT M1636

27. UEFI VARIABLE DELETION/CORRUPTION

Description:

Input validation error in MinnowBoard 3 Firmware versions prior to 0.65 allow local attacker to cause denial of service via UEFI APIs.

Recommendation:

This update improves input validation by firmware and is strongly recommended.

For firmware development projects, incorporate the updates in https://github.com/tianocore/edk2-platforms/tree/devel-MinnowBoard3-UDK2017

When using MinnowBoard 3, update to version 0.65 or later. Updated firmware is available at https://firmware.intel.com/projects/minnowboard3

Acknowledgments:

Reported by Intel.

References:

CVE-2017-5699