TCG Platform Reset Attack Mitigation Specification

Specification Version 1.00
Revision 1.00
May 15, 2008
Published

Contact: admin@trustedcomputinggroup.org

TCG Published

TCG

Revision History

r1.0	**Published**

Table of Contents

1 Introduction

Start of informative comment:

Theory of Operation

When a platform reboots or shuts down, the contents of volatile memory (RAM) are not immediately lost. Without an electric charge to maintain the data in memory, the data will begin to decay. During this period, there is a short timeframe during which an attacker can turn off or reboot the platform, and quickly turn it back on to boot into a program that dumps the contents of memory. Encryption keys and other secrets can be easily compromised through this method.

Host Platform Reset threats to the S-CRTM can be mitigated by a BIOS-initiated system memory operation that overwrites system memory on the next platform reboot. The BIOS must overwrite memory with information unrelated to the secrets in memory that may be exposed to an attacker after a Host Platform reset; zeroing memory is just one example of an effective memory overwrite operation.

The BIOS is not required to initiate and complete the memory overwrite operation on every platform reboot, but is required to initiate and complete a memory overwrite operation every time it is signaled to do so by the OS using a bit setting in Host Platform non-volatile memory. In this specification, this bit setting, which persists across all types of Host Platform Resets, is called *the Memory Overwrite Request (MOR) bit*. Figure 1 and Figure 2, which are part of this Informative comment, show how BIOS POST, the IPL code, and the OS use the MOR bit to communicate with each other across all types of Host Platform Reset events.

A general description of the scheme is that after any type of Host Platform Reset event (with the exception of a CPU-only reset that is used by some chipsets to turn off a CPU feature without re-setting other Host Platform components), if signaled to do so by the OS, the POST BIOS must, prior to executing any non-BIOS code, overwrite system memory.

Figure 1 shows the sequence where BIOS POST reads the MOR bit before BIOS POST executes any Option ROM code, the IPL code sets the MOR bit before the IPL code puts any secrets in the clear in system memory, and the OS clears the MOR bit across a Host Platform Reset event that includes a controlled OS shutdown; in this case the BIOS POST code does not initiate a memory overwrite operation during the next Host Platform boot operation.

In Figure 2, a controlled OS shutdown is not part of the Host Platform Reset event, so this a potential attack. Comparing Figure 2 with Figure 1 shows that because the OS shutdown code was not executed, the OS did not clear the non-volatile MOR bit. When BIOS POST code executes the next time the Host Platform reboots it will read a ‘1’ from the MOR bit, so it must initiate a vendor-specific method that overwrites all of system memory and the processor caches. When that memory clear operation completes successfully, BIOS POST clears the MOR bit.

Note that in Figure 2, although the box labeled “Memory Overwrite method” is shown outside the BIOS POST code execution arrow, this does not mean execution of this method affects PCR contents; the code that either initiates a hardware-assisted method of overwriting memory or the code that overwrites memory without hardware assistance is a block of code in BIOS POST, which is already measured into a PCR.

In Figure 1,and in the part of Figure 2 that shows a controlled OS shutdown, the IPL code writes a ‘1’ to the MOR flag before it has any secrets in system memory to protect. Figure 1 also shows that subsequently, as part of the controlled OS shutdown, the OS writes a ‘0’ to the MOR bit when there are no more secrets in memory to protect. Between these two OS-initiated write events to the MOR bit, the OS protects the secrets in system memory.

Note: If the Host Platform boots into an OS that does not implement the functionality discussed in this specification, the MOR bit is never set.

Because clearing memory may be required for reasons and platform functions not related to the MOR bit or purposes and threats associated with this specification or any other TCG operation, nothing in this specification prohibits any memory clear operation.

Scope, Security and Trust Assumptions

The scope of this specification applies only to the contents of memory under control of the Operating System within the Static RTM, however, it's possible (and even likely) memory under control of a D-RTM is also cleared as a result of the methods described in this specification.

The attacks mitigated by the methods in this specification are limited to simple rebooting of the system. An example of an attack to be mitigated is a stolen PC Client which is in a suspended state. After several unsuccessful attempts to guess the OS lock password, the attacker forces the platform to reboot without shutting down the OS and reboots to a CD containing an attack OS from which the attacker expects to read the contents of memory. The methods in this specification are not intended to protect against active physical attacks beyond the scope of the above scenario.

All secrets capable of being cleared by the methods in this specification are exposed to Ring 0, therefore, the protections to invoke and control these methods are also exposed to Ring 0. For this reason, this specification makes a fundamental assumption that the Operating System protects Ring 0 and any violation of those protections renders the methods discussed in this specification useless.

The security assurance level provided by the MOR bit mechanism is strengthened if the data to be protected is sealed using PCR [0].

Adding the functionality that is in this specification to the BIOS does not open the BIOS up to additional attacks.

If an attacker crashes the OS with physical access, as long as BIOS has not been replaced since the last boot cycle, the secrets in system memory are still protected. If a Host Platform has an OS environment for BIOS update, that platform may be at risk. This specification assumes the Host Platform manufacturer tightly controls BIOS update. The requirements for protecting the BIOS update process are stated in the relevant PC Client Specifications and the TBB Protection Profiles.

Functionality

The IPL code uses an INT 1Ah function offered by the BIOS POST code to set the MOR bit before the IPL code puts any secrets in the clear in system memory. This INT 1Ah function is defined in Section 4.

For a UEFI boot, the EFI OS loader uses the MemoryOverwriteRequestControl EFI variable to set the MOR bit prior to the loader putting any secrets in the clear in system memory. This variable is defined in Section 5.

OS code uses a function in the ACPI _DSM control method in the TPM 1.2 ACPI device object to clear the MOR bit as part of a controlled OS shutdown. This _DSM control method function is defined in Section 6. The requirements in that section are based on the industry-standard ACPI 3.0 specification. The BIOS persists the result of the OS calling this _DSM method function across Host Platform reboots by using a vendor-specific bit in a non-volatile storage location on the Host Platform.

On a UEFI system, the MemoryOverwriteRequestControl EFI variable described in Section 5 can be updated to clear the MOR bit after secrets have been removed from memory.

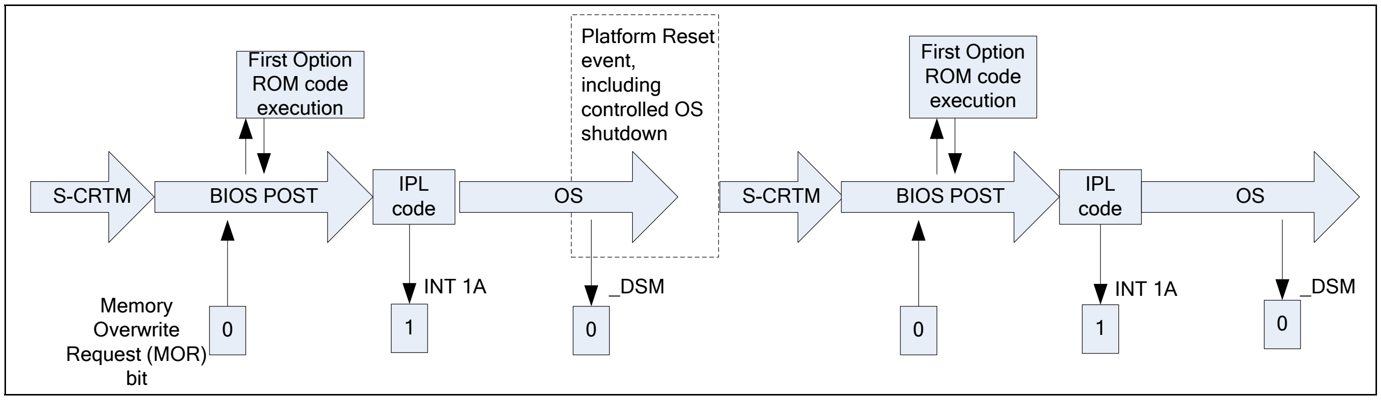
First Option ROM code execution
Platform Reset event, including controlled OS shutdown
First Option ROM code execution
S-CRTM
BIOS POST
IPL code
OS
S-CRTM
BIOS POST
IPL code
OS
Memory Overwrite Request (MOR) bit
0
INT 1A
1
_DSM
0
0
INT 1A
1
_DSM
0

Figure 1 Platform Boot Cycle with Complete OS Shutdown

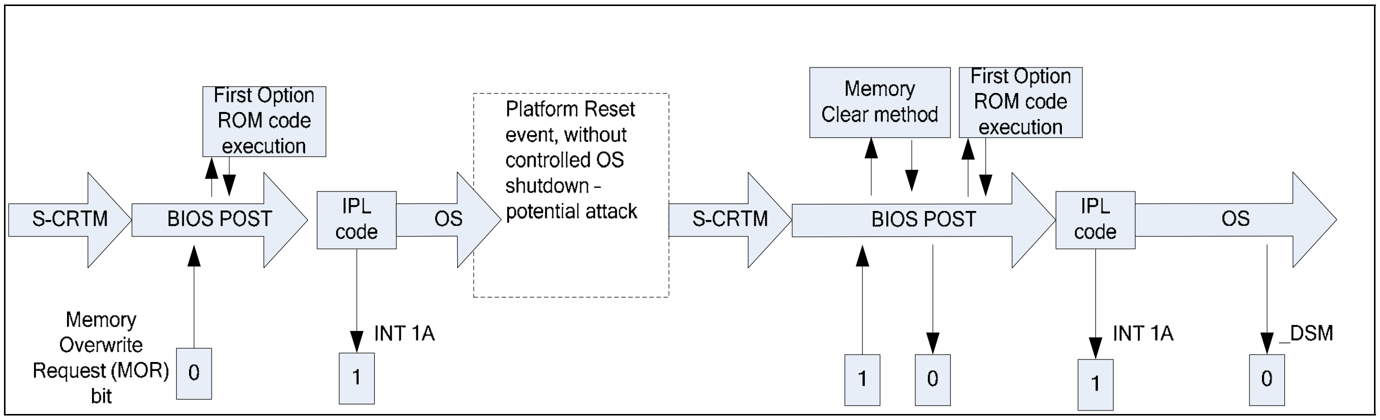
First Option ROM code execution
Platform Reset event, without controlled OS shutdown – potential attack
Memory Clear method
First Option ROM code execution
S-CRTM
BIOS POST
IPL code
OS
S-CRTM
BIOS POST
IPL code
OS
Memory Overwrite Request (MOR) bit
0
INT 1A
1
1
0
INT 1A
1
_DSM
0

Figure 2 Platform Boot Cycle: without Complete OS Shutdown

End of informative comment.

2 Requirements

Start of informative comment:

This section contains all the mandatory requirements for this specification for clearing memory upon unexpected resets and reboots.

End of informative comment.

2.1 General Requirements

Start of informative comment:

During S1 -> S3, the operating system is expected to retain secrets in memory and not rely on protections provided by the MOR bit. The BIOS, therefore, takes no action entering or leaving any of these operational states. When entering S4 and S5, the operating system depends on the protections provided by the MOR bit and therefore the BIOS is expected to honor the MOR bit. The BIOS is expected to detect and take action on the MOR bit upon resuming from these operational states.

Item 3.b below requires the BIOS to attempt to detect any potential tampering with the MOR bit. Tampering of the MOR bit could cause the BIOS, upon reset, to ignore a necessary memory clear operation. However, because data can only be protected by a TPM when the TPM has an owner, the need to clear memory applies only when there is a TPM owner. Checking the TPM for ownership prevents unnecessarily extended boot times in cases where it is not possible for the OS to have sealed data to a PCR, therefore it is only necessary to check if there is an owner if and only if the MOR bit is set. Examples of these situations are: the manufacturing floor, prior to OS installation, or if the OS does not make use of the TPM's boot time data protection capabilities.

End of informative comment.

1. BIOS POST MUST support reading and writing the Memory Overwrite Request (MOR) bit to and from non-volatile storage on the Host Platform
2. To enable IPL code to communicate MOR bit settings to POST BIOS, POST BIOS MUST support the INT 1Ah, sub-function BBh, interface function TCG_SetMemoryOverwriteRequestBit.
3. If there is an owner installed in the TPM and either of the following conditions occur, the BIOS MUST initiate the process that clears all system memory and the processor caches:
    a. The BIOS detects the MOR bit is set, or
    b. The BIOS detects any reliability or integrity issue with NVM on the Host Platform.
4. The MOR request (i.e., checking of MOR bit and memory clear operation) SHOULD be performed before control is transferred outside of the S-CRTM, and MUST be performed before IPL, option ROM or other arbitrary code can be executed.
5. The BIOS MAY perform a memory clear operation for reasons unrelated to the MOR bit or for purposes and threats not associated with this specification.

2.2 Memory Overwrite Request Optimizations

Start of informative comment:

In order to ensure that the memory overwrite process is performed as efficiently as possible, system builders should be aware of additional design considerations. If the MOR request is performed too early in the boot process, the system may not be able to take advantage of the full speed of memory. This may greatly increase the time required to overwrite memory, and can result in slower boot times and increased user confusion. Ideally, the MOR request should be performed as soon as memory has been initialized and can be overwritten with minimal clock cycles per byte.

UEFI platform firmware can use known art to ensure that flash wear-leveling occurs in the UEFI variable store since this MemoryOverwriteRequestControl variable will be written possibly two times per platform boot.

End of informative comment.

2.3 Auto Detection of Clean Static RTM OS Shutdown

Start of informative comment:

Some Static RTM Operating Systems may not clear the MOR bit prior to shutting down. This may cause the BIOS to always perform a memory clear operation. While not a security concern, this will cause unnecessary delays in the platform's boot process. Static RTM Operating Systems that do not clear the MOR bit upon a clean shutdown are encouraged to provide an option to allow the platform owner to opt-out of the protections provided by the MOR. These operating systems may also indicate this potential behavior by setting the DisableAutoDetect bit to zero in the MemoryOverwriteAction_BitValue parameter. When this bit is zero, the BIOS is permitted to detect a clean shutdown of the Static RTM Operating System and clear the flag itself.

There are various target shutdown operational states and certain conventional steps a Static RTM Operating System takes when transitioning to those states. If allowed by the DisableAutoDetect bit, the BIOS may trigger on an action taken during these shutdown steps provided the trigger occurs after the secrets have been cleared by the Static RTM Operating System. Known operational state transitions and their triggers are identified in the normative sections below. The most common method for clearing the MOR upon detecting one of these triggers is the use of SMI but no particular method is mandated by this specification.

Static RTM Operating Systems that always clear the MOR bit upon a clean shutdown (i.e., they will always call the MOR interface) will set the DisableAutoDetect bit to the value 1 indicating to the BIOS that it should not automatically detect the Static RTM's Operating System's shutdown.

Static RTM Operating Systems that allow the BIOS to autodetect a clean shutdown must ensure that secrets are cleared from memory prior to any triggering event listed below.

The descriptions below are provided as example implementation to auto-detect an orderly shutdown of the Operating System.

Use of ACPI to auto-detect an orderly shutdown of the Operating System:

The following is an excerpt from the ACPI Specification to provide context:

If ACPI platform supports both ACPI and legacy modes of operation, it must contain a HW register capable of generating a HW event. In case of PC this event is SMI.

OSPM uses this register to make the hardware switch in and out of ACPI mode. Within the FADT there are three values that signify the address (SMI_CMD) of this port and the data value written to enable the ACPI state (ACPI_ENABLE), and to disable the ACPI state (ACPI_DISABLE).

To transition an ACPI/Legacy platform from the Legacy mode to the ACPI mode the following would occur:

- ACPI driver checks that the SCI_EN bit is zero, and that it is in the Legacy mode.
- OSPM does an OUT to the SMI_CMD port with the data in the ACPI_ENABLE field of the FADT.
- OSPM polls the SCI_EN bit until it is sampled as SET.

To transition an ACPI/Legacy platform from the ACPI mode to the Legacy mode the following would occur:

- ACPI driver checks that the SCI_EN bit is one, and that it is in the ACPI mode.
- OSPM does an OUT to the SMI_CMD port with the data in the ACPI_DISABLE field of the FADT.
- OSPM polls the SCI_EN bit until it is sampled as RESET.

End of ACPI Specification excerpt.

Platforms typically boot to OS in legacy mode and then are switched to ACPI mode of operation by OS executing the ACPI_ENABLE command. In the case of a normal shut down the OS executes the ACPI_DISABLE command.

BIOS requests any or both of the above commands simply by populating corresponding fields in the FADT table and thereby supplying this to the SMI handler code.

Each of the above commands leads to SMI being generated and may along with other tasks operate MOR bit – assert it in ACPI_ENABLE SMI and de-assert it in ACPI_DISABLE SMI

This method assumes that ACPI_DISABLE is in the common path for both system reset and shut down. It is preferable if by the time ACPI_DISABLE SMI is generated secrets are already erased from memory.

An alternate example to auto-detect an orderly shutdown of the Operating System:

If the chip set supports it, trap on an IO to the operating system's reset register, for example:

1> Determine the register that the OS accesses to cause a restart.

2> Pick a point in the restart path where the OS turns control over to the BIOS (For example, use a _REG on the LPC bus of the ACPI name space.) This is done to prevent large numbers of SMIs due to PS/2 mouse/keyboard traffic. At this point, turn on IO trapping.

3> Examine subsequent SMIs for the restart value written to the restart register. If this is the restart scenario, clear the MOR bit and then allow the restart to occur.

End of informative comment.

It is permissible for system firmware to be implemented such that it automatically clears MOR on detection of an orderly shutdown of the OS. Determination of an orderly shutdown of the OS is OS and firmware specific.

3 MemoryOverwriteAction_BitValue Values

Start of informative comment:

Values for MemoryOverwriteAction_BitValue Parameter which are common to both Convention and UEFI functions.

End of informative comment.

Table 1 Variable Layout

Mnemonic	Bit Offset	Bit Length	Description
ClearMemory	0	1	0 = Firmware MUST clear the MOR bit 1 = Firmware MUST set the MOR bit
Reserved	1	3	Reserved (currently unused, Caller MUST set all to 0s)
DisableAutoDetect	4	1	0 = Firmware MAY autodetect a clean shutdown of the Static RTM OS. See section 2.3 Auto Detection of Clean Static RTM OS Shutdownfor details. 1=Firmware MUST NOT autodetect a clean shutdown of the Static RTM OS
Reserved	5	3	Reserved (currently unused, Caller MUST set all to 0s)

4 Interface for Conventional BIOS

4.1 TCG_SetMemoryOverwriteRequestBit Function

INT 1Ah, (AH)=BBh, (AL)=08h

This function sets or clears the Memory Overwrite Request (MOR) bit.

On entry:

(AH)	=	BBh
(AL)	=	08h
(ES)	=	Segment portion of pointer to input parameter block
(DI)	=	Offset portion of pointer to input parameter block
(EBX)	=	41504354h
(ECX)	=	0
(EDX)	=	0

On return:

(EAX)	=	Return Code as defined in the section titled “Application Level Interface – INT 1A TCG Functions” in the latest TCG PC Client Specific Implementation Specification for Conventional BIOS.

All other registers are preserved.

4.2 TCG_SetMemoryOverwriteRequestBit Input Parameter Block

Offset	**Size**	**Field Name**	**Description**
00h	WORD	IPBLength	The length, in bytes, of the input parameter block, set to 0005h.
02h	WORD	Reserved	Reserved for future definition by this specification. Caller MUST set all to 0s
04h	BYTE	MemoryOverwriteAction_BitValue	This parameter sets The value POST BIOS is to set the non-volatile Memory Overwrite Request (MOR) bit to. Values are described in Table 1 Variable Layout.

Table 2 TCG_SetMemoryOverwriteRequestBit Input Parameter Block

5 Interface for UEFI

Start of informative comment:

UEFI uses a variable rather than a callable interface to set and clear the MOR bit.

End of informative comment.

5.1 UEFI Variable

5.1.1 The MemoryOverwriteRequestControl

Start of informative comment:

The purpose of the MemoryOverwriteRequestControl UEFI variable is to give users (e.g., OS, loader) the ability to indicate to the platform that secrets are present in memory and that the platform firmware must clear memory upon a restart.

The OS loader should not create the variable. Rather, the firmware is required to create it and must support the semantics described here.

End of informative comment.

5.1.2 GUID

```
#define MEMORY_ONLY_RESET_CONTROL_GUID \
     {0xe20939be, 0x32d4, 0x41be, 0xa1, 0x50, 0x89, 0x7f, 0x85, 0xd4, 0x98,
     0x29}
```

5.1.3 Description

The name of the new UEFI variable will be "**MemoryOverwriteRequestControl**" and it is a 1 byte unsigned value. The attributes should be:

```
    EFI_VARIABLE_NON_VOLATILE |
    EFI_VARIABLE_BOOTSERVICE_ACCESS |
    EFI_VARIABLE_RUNTIME_ACCESS
```

since the variable needs to hold across all types of reboots, and be available at boot time and run time.

The layout of the variable is described in Table 1 Variable Layout.

5.2 Usage

Variable creation: Upon each reboot, the platform firmware must check for the existence and correct attributes of this variable. If the variable does not exist as defined above, and the TPM is enabled, the platform firmware must create the variable as defined above. If at this time, there is no TPM owner, the platform firmware should set the initial value of the variable to 0. If there is a TPM owner, the platform must set the initial value of the variable to 1.

Upon each reboot, if there is a TPM owner established, the platform firmware must check the value of bit 0. If bit 0 is set, the platform firmware must clear all memory prior to continuing with the boot process. Once the memory is cleared, bit zero should be cleared since any secrets have been removed.

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

System software is expected to manage secrets in memory and manipulate the state of the variable such that BIOS will not normally be required to clear memory.

If a call to SetVariable with the correct GUID and variable name (MemoryOverwriteRequestControl) either has *DataSize* set to 0, or the *Attributes* value does not match the *Attributes* value returned from GetVariable, the service must return EFI_INVALID_PARAMETER without changing the state of the variable.

Some of the platform flows are shown in Figure-3 and Figure-4.

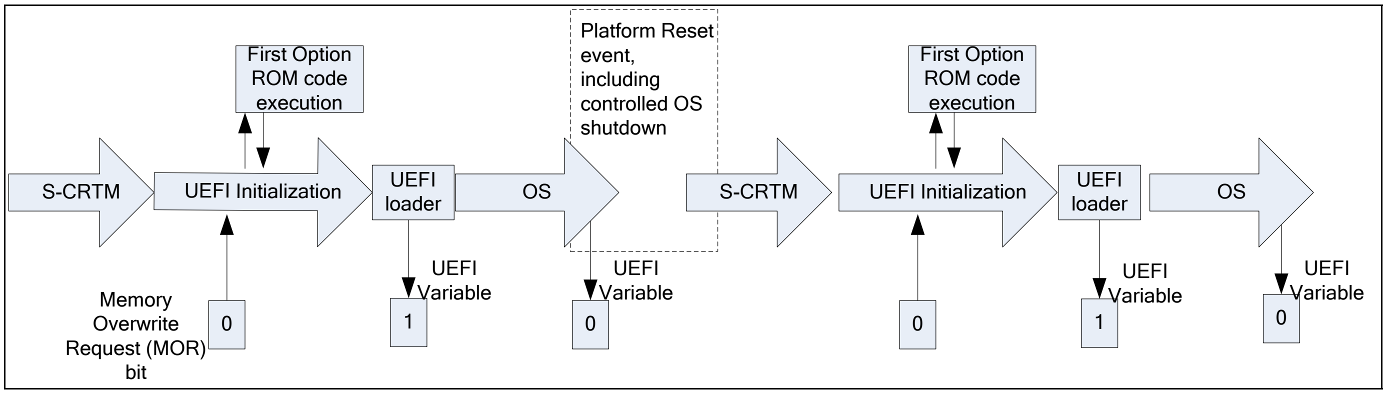
Platform Reset event, including controlled OS shutdown
First Option ROM code execution
S-CRTM
UEFI Initialization
UEFI loader
OS
UEFI Variable
Memory Overwrite Request (MOR) bit
0
1
0
S-CRTM
UEFI Initialization
First Option ROM code execution
UEFI loader
OS
UEFI Variable
UEFI Variable
0
1
0

Figure 3 UEFI Platform Boot Cycle with Complete OS Shutdown

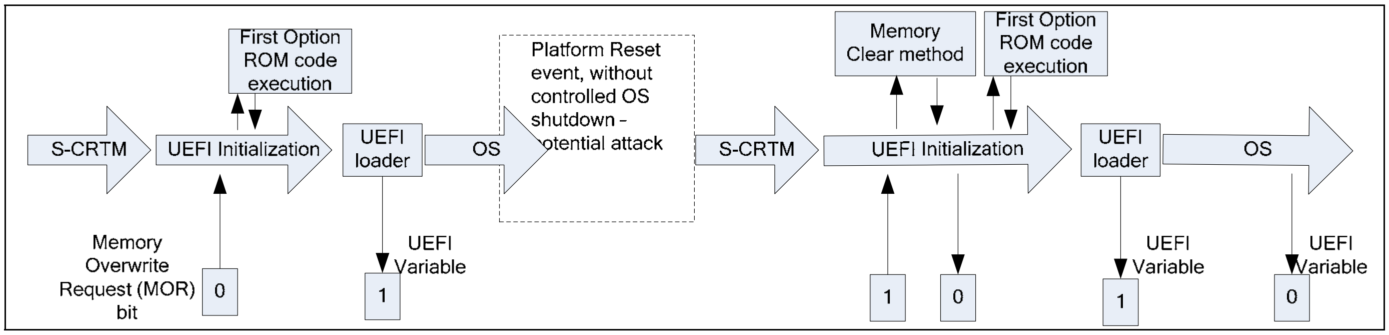
First Option ROM code execution
Platform Reset event, without controlled OS shutdown – potential attack
Memory Clear method
First Option ROM code execution
S-CRTM
UEFI Initialization
UEFI loader
OS
S-CRTM
UEFI Initialization
UEFI loader
OS
Memory Overwrite Request (MOR) bit
0
UEFI Variable
1
1
0
UEFI Variable
1
UEFI Variable
0

Figure 4 UEFI Platform Boot Cycle: Without Complete OS Shutdown

6 ACPI _DSM Function

OS code MUST use a _DSM ACPI control method to clear the MOR bit as part of a controlled OS shutdown.

- Table 2 defines the function that MUST be exposed by the BIOS, and the behavior the OS can expect upon invoking this function.
- The function MUST reside in the _DSM control method in the ACPI device object for the TPM 1.2. The UUID function identifier to be used exclusively for the Memory Clear Interface MUST be “376054ED-CC13-4675-901C-4756D7F2D45D”
- Function indices MUST start at index 1, since function 0 is the standard _DSM query function

Table 3 Memory Clear Interface Functions

Function Definition	**Function Behavior**
a) Set MOR Bit State <u>Arguments:</u> Arg0 (Buffer): UUID = 376054ED-CC13-4675-901C-4756D7F2D45D Arg1 (Integer): Revision ID = 1 Arg2 (Integer): Function Index = 1 Arg3 (Package): Arguments = Package -- Type: Integer Purpose: Operation Value of the Request Description: Byte 0 of Arg3 is defined in Table 1 Variable Layout The caller MUST set all other bytes of Arg3 to 0. <u>Returns:</u> Type: Integer Purpose: Function return code Description: 0: Success 1: General Failure <u>Example:</u> A submitted operation value of 0 and a return value of 0 indicates that the MOR bit has been cleared.	This function allows the OS to force the BIOS to initiate a memory overwrite operation on the next boot cycle. If the OS calls this function with Arg3.ClearMemory = 1 and the function successfully sets the MOR bit, then the function MUST return Success. If the OS calls this function with Arg3.ClearMemory = 1 and the function is unable to set the MOR bit, then the function MUST return General Failure If the OS calls this function with Arg3.ClearMemory = 0 and the function successfully clears the MOR bit, then the function MUST return Success. If the OS calls this function with Arg3.ClearMemory = 0 and the function is unable to clear the MOR bit, then the function MUST return General Failure.