# rXX TCG MOR Test

### XX.1TCG MOR Test

Reference Document:

*TCG PC Platform Reset Attack Mitigation Specification*,v1.10 Revision 17, January 21, 2019, Chapter 4.

These tests support platform firmware that implements the MemoryOverwriteRequestControl and MemoryOverwriteRequestControlLock UEFI variables in accordance with TCG PC Platform Reset Attack Mitigation Specification, v1.10.

### XX.1.1 MemoryOverwriteRequestControl Platform Reset Check

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 1 | { 0xd1c31d5f, 0x944f, 0x434c, {0x88, 0x26, 0xa6, 0xfb, 0x00, 0xd4, 0xbb, 0x97 }} | Verify MemoryOverwriteRequestControl is created by platform firmware as specified in the TCG specification. | 1. Reset the system.  2. Verify GetVariable() returns MemoryOverwriteRequestControl with correct attributes = (NV+RT+BS) and correct DataSize.  If MemoryOverwriteRequestControl does not exist with the correct attributes and/or DataSize, the test should report a failure. |
| 2, 3 | { 0xa6c191b1, 0x9bfb, 0x4231, {0xb9, 0x2d, 0x4e, 0xa2, 0x21, 0xf6, 0x4b, 0x5d }},  { 0x34d41a18, 0x8ffe, 0x456b, {0xa4, 0x82, 0x97, 0xb9, 0x9c, 0x50, 0x7d, 0x9e }} | Set the MemoryOverwriteAction\_BitValue within MemoryOverwriteRequestControl.  Verify that on System reset, the MemoryOverwriteAction\_BitValue is cleared. | 1. Invoke SetVariable() with valid GUID, variable name, attributes, and DataSize and Set MemoryOverwriteAction\_BitValue Bit 0.  2. Reboot System.  3. Invoke GetVariable() with valid GUID, variable name, attributes, and DataSize.  4. Verify MemoryOverwriteAction\_BitValue Bit 0 is Cleared. |
| 4 | { 0x002519c6, 0x859f, 0x4d25, {0xb1, 0x36, 0xb2, 0xef, 0x61, 0xe5, 0xd9, 0x6f }} | Verify MemoryOverwriteRequestControlLock is created by platform firmware as specified in the TCG specification. | 2. Verify GetVariable() returns EFI\_SUCCESS and MemoryOverwriteRequestControlLock with correct attributes = (NV+RT+BS), correct DataSize = 1, and correct Data = 0x00 to indicate unlocked. |

### XX.1.2 MemoryOverwriteRequestControl.SetVariable()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5, 6 | { 0x13e97668, 0xe905, 0x452d, {0xa0, 0x72, 0x03, 0xde, 0xbf, 0x53, 0xdf, 0xa2 }},  { 0x94811e97, 0x7513, 0x4099, {0xae, 0xb8, 0x5a, 0x7e, 0x7d, 0x50, 0xb4, 0x99 }} | MemoryOverwriteRequestControl.SetVariable() returns EFI\_INVALID\_PARAMETER by passing in all valid parameters aside from an incorrect DataSize = 0.  The state of MemoryOverwriteRequestControl shall not change. | 1. Invoke GetVariable()with valid GUID, variable name, attributes and DataSize. Save the current value for comparison.  2. Invoke SetVariable() with valid GUID, variable name, and attributes. DataSize set to 0 and Data is != saved value from the previous step.  a. Verify SetVariable() returns EFI\_INVALID\_PARAMETER.  b. Verify that MemoryOverwrit-eRequestControl GetVariable() returns a Data value == initial saved value of MemoryOverwriteRequestControl. |
| 7, 8 | { 0x15378290, 0x17dd, 0x4cbf, {0x8e, 0x8f, 0xc2, 0x4b, 0xc9, 0x09, 0xb4, 0x14 }},  { 0x5f0b42ef, 0x3cec, 0x46ad, {0xbc, 0x5b, 0xc3, 0x7a, 0x1d, 0x45, 0x41, 0x29 }} | MemoryOverwriteRequestControl.SetVariable() returns EFI\_INVALID\_PARAMETER by passing in all valid parameters aside from incorrect attributes.  The state of MemoryOverwriteRequestControl shall not change. | 1. Invoke GetVariable()with valid GUID, variable name, attributes and DataSize. Save the current value for comparison.  2. Invoke SetVariable() with valid GUID, variable name, and DataSize = 1. Pass an attributes value != (NV+RT+BS) and Data != saved value from the previous step.  a. Verify SetVariable() returns EFI\_INVALID\_PARAMETER.  b. Verify that MemoryOverwrit-eRequestControl GetVariable() returns a Data value == initial saved value of MemoryOverwriteRequestControl. |

### XX.1.3 MemoryOverwriteRequestControlLock.SetVariable()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 9, 10 | { 0x9665c8d9, 0x267c, 0x4393, {0xa4, 0x72, 0x10, 0x54, 0x43, 0x56, 0x20, 0x21 }},  { 0xce655812, 0xcc40, 0x42b0, {0x80, 0xd1, 0xa7, 0x26, 0xdb, 0x14, 0x9e, 0xa8 }} | MemoryOverwriteRequestControlLock.SetVariable() passing valid parameters excluding Attributes == 0 must return EFI\_WRITE\_PROTECTED.  The state of MemoryOverwriteRequestControlLock shall not change. | 1. Reboot the system to ensure MemoryOverwriteRequestControlLock is unlocked.  2. Invoke SetVariable() with valid GUID, variable name, Data, and DataSize = 1. Pass an attributes value == 0.   1. Verify SetVariable() returns EFI\_WRITE\_PROTECTED. 2. Verify that MemoryOverwriteRequestControlLock GetVariable() returns a Data Value == 0x00. |
| 11, 12 | { 0xd34348ad, 0x5d46, 0x4961, {0x91, 0x3e, 0xb1, 0xf2, 0xc5, 0xe7, 0x0f, 0x7d }},  { 0x838311e0, 0x419a, 0x4c92, {0x90, 0x60, 0xbf, 0x94, 0x59, 0xd3, 0xc3, 0x59 }} | MemoryOverwriteRequestControlLock.SetVariable() passing valid parameters excluding DataSize == 0 must return EFI\_WRITE\_PROTECTED.  The state of MemoryOverwriteRequestControlLock shall not change. | Invoke SetVariable() with valid GUID, variable name, Data, and Attributes. Pass DataSize== 0.   1. Verify SetVariable() returns EFI\_WRITE\_PROTECTED. 2. Verify that MemoryOverwriteRequestControlLock GetVariable() returns a Data Value == 0x00. |
| 13, 14 | { 0x12998a9c, 0xc863, 0x4572, {0x80, 0x6c, 0xb6, 0x40, 0x97, 0x91, 0x89, 0x7f }},  { 0xebd4ba26, 0x44c3, 0x464a, {0x88, 0xe3, 0x3b, 0x44, 0x94, 0x88, 0xbb, 0xb8 }} | MemoryOverwriteRequestControlLock.SetVariable() passing valid parameters excluding Data == NULL must return EFI\_WRITE\_PROTECTED.  The state of MemoryOverwriteRequestControlLock shall not change. | Invoke SetVariable() with valid GUID, variable name, Attributes, and DataSize = 1. Pass Data == NULL.   1. Verify SetVariable() returns EFI\_WRITE\_PROTECTED. 2. Verify that MemoryOverwriteRequestControlLock GetVariable() returns a Data Value == 0x00. |
| 15, 16 | { 0x00920c8e, 0x241d, 0x4da0, {0x8b, 0x59, 0xc3, 0xad, 0xe0, 0xd6, 0x4e, 0xc9 }},  { 0x169a43cc, 0x23fa, 0x4887, {0x8b, 0x4f, 0x11, 0xb2, 0xa0, 0x88, 0x4f, 0x63 }} | MemoryOverwriteRequestControlLock.SetVariable() passing valid parameters and Attributes != (NV+BS+RT) must return EFI\_INVALID\_PARAMETER.  The state of MemoryOverwriteRequestControlLock shall not change. | Invoke SetVariable() with valid GUID, variable name, Data, and DataSize. Set Attributes != (NV+BS+RT).   1. Verify SetVariable() returns EFI\_INVALID\_PARAMETER. 2. Verify that MemoryOverwriteRequestControlLock GetVariable() returns a Data Value == 0x00. |
| 17, 18 | { 0x7c11d0ab, 0x7a84, 0x482b, {0xb7, 0x5a, 0xd7, 0x72, 0x6c, 0x3e, 0xae, 0x0e }},  { 0xbbaa52b6, 0x5576, 0x4acd, {0xb9, 0x5f, 0x86, 0x22, 0x5e, 0xfc, 0xb0, 0x31 }} | MemoryOverwriteRequestControlLock.SetVariable() passing valid parameters excluding (DataSize != 1 && DataSize != 8) must return EFI\_INVALID\_PARAMETER.  The state of MemoryOverwriteRequestControlLock shall not change. | Invoke SetVariable() with valid GUID, variable name, Data, and Attributes. Set DataSize != 1 && DataSize != 8.   1. Verify SetVariable() returns EFI\_INVALID\_PARAMETER. 2. Verify that MemoryOverwriteRequestControlLock GetVariable() returns a Data Value == 0x00. |

### XX.1.4 MemoryOverwriteRequestControlLock Unlocked state

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 19, 20 | { 0x60b227e0, 0x9f5c, 0x412c, {0x88, 0x79, 0xd3, 0x59, 0xfc, 0xd0, 0xdd, 0x3f }},  { 0x2eba284a, 0xf701, 0x4c19, {0xbe, 0x5c, 0x39, 0x27, 0xb0, 0x68, 0x4f, 0xd7 }} | Setting MemoryOverwriteRequestControlLock to Unlocked State when already Unlocked should using SetVariable should return EFI\_SUCCESS and a data value of 0(Unchanged). | 1. Reboot System to make firmware initialize MemoryOverwriteRequestControlLock to Unlocked.  2. Invoke SetVariable() with Valid GUID, variable name, attributes, DataSize = 1, and Data = 0x00.   1. Verify Lock.SetVariable() returns EFI\_SUCCESS.   3. Invoke GetVariable() with valid GUID, variable name, attributes, DataSize.   1. Verify that the output value is 0x00. |
| 21, 22 | { 0x9a1b44ae, 0x08ce, 0x474c, {0xa5, 0x8e, 0xa6, 0xe2, 0xcf, 0xaf, 0x91, 0x2c }},  { 0x490d7b39, 0xcad4, 0x4e8c, {0xb1, 0x5d, 0x63, 0xd2, 0x0c, 0xb3, 0xe9, 0x45 }} | Setting MemoryOverwriteRequestControlLock with an invalid first byte for Data when in Unlocked State should return EFI\_INVALID\_PARAMETER. | 1. Invoke SetVariable() with valid GUID, variable name, attributes, DataSize = 1, Data first byte != 0 && Data first byte != 1.   1. Verify SetVariable() returns EFI\_INVALID\_PARAMETER   2. Invoke GetVariable() with valid GUID, variable name, attributes, DataSize.   1. Verify that the output value is 0x00. |

### XX.1.5 MemoryOverwriteRequestControlLock Locked w/o key state

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 23, 24 | { 0x397394ae, 0xce01, 0x4350, {0xa2, 0x0c, 0xe1, 0xb3, 0xe8, 0x74, 0xdc, 0x01 }},{ 0x9cf27a60, 0x94b5, 0x4e2e, {0xb3, 0x2b, 0x51, 0x0f, 0x24, 0x7a, 0x80, 0xd7 }} | Changing MemoryOverwritRequestControlLock to Locked state without key Should return EFI\_SUCCESS when MemoryOverwriteRequestControlLock is currently unlocked. | 1. Reboot System to make firmware initialize MemoryOverwriteRequestControlLock to Unlocked.  2. Invoke SetVariable() with Valid GUID, variable name, attributes, DataSize, and Data Input = 0x01.   1. Verify SetVariable() returns EFI\_SUCCESS. 2. Verify GetVariable() returns Data value of 0x01. |
| 25, 26 | { 0x766dc008, 0x2a88, 0x4eed, {0x91, 0x95, 0x46, 0x92, 0xdc, 0xcc, 0x1d, 0xf6 }}, { 0x5f66c8e5, 0x1bf8, 0x4af4, {0x86, 0x45, 0xf4, 0x93, 0xa0, 0xee, 0x26, 0x88 }} | Changing MemoryOverwriteRequestControlLock to Unlocked state Should return  EFI\_ACCESS\_DENIED when MemoryOverwriteRequestControlLock is currently Locked without key. | 1. Invoke SetVariable() with Valid GUID, variable name, attributes, DataSize, and Data Input = 0x00.   1. Verify SetVariable() returns EFI\_ACCESS\_DENIED. 2. Verify GetVariable() returns Data value of 0x01. |
| 27, 28 | { 0x1064f6ce, 0xb307, 0x4981, {0xac, 0x8f, 0xe5, 0xca, 0x20, 0x6c, 0x1b, 0x8b }}, { 0x40f2c4e9, 0xe937, 0x426e, {0x98, 0xc5, 0x62, 0xca, 0x23, 0x68, 0x52, 0xd0 }} | Changing MemoryOverwriteRequestControlLock to Locked without key state Should return EFI\_ACCESS\_DENIED when MemoryOverwriteRequestControlLock is currently Locked without key. | 1. Invoke SetVariable() with Valid GUID, variable name, attributes, DataSize, and Data Input = 0x01.   1. Verify SetVariable() returns EFI\_ACCESS\_DENIED. 2. Verify GetVariable() returns Data value 0x01. |
| 29, 30 | { 0x6d5dcb4d, 0xe008, 0x41cc, {0x98, 0x32, 0xdf, 0xa8, 0x38, 0xb1, 0xb6, 0x02 }}, { 0x29b81ee0, 0x368d, 0x447c, {0x9e, 0xd0, 0xa5, 0xfe, 0xd7, 0x02, 0x65, 0x7b }} | Changing MemoryOverwriteRequestControlLock to Locked state with key  Should return EFI\_ACCESS\_DENIED when MemoryOverwriteRequestControlLock was already Locked without key. | 1. Invoke SetVariable() with Valid GUID, variable name, attributes, DataSize, and Data Input = 8-byte user generated key.   1. Verify SetVariable() returns EFI\_ACCESS\_DENIED. 2. Verify GetVariable() returns Data value 0x01. |
| 31, 32 | { 0xe88b22ce, 0x3de6, 0x49b0, {0xb2, 0x46, 0x9e, 0x35, 0x98, 0x2b, 0x9b, 0x1c }}, { 0xc8894201, 0x7a3a, 0x47d4, {0xa8, 0x8d, 0xdf, 0x4b, 0x03, 0xff, 0xde, 0x4f }} | Since the MemoryOverwriteRequestControlLock is Set, Changing MemoryOverwriteRequestControl value should return EFI\_ACCESS\_DENIED and the value of MemoryOverwriteRequestControl Should not change | 1. Invoke MOR - GetVariable()with valid GUID, variable name, attributes and DataSize. Save the current value for comparison.  2. Invoke MOR - SetVariable() with valid GUID, variable name, and DataSize. Pass Data value with Bit0 Set High to indicate MORbit0 set.  a. Verify SetVariable() returns EFI\_ACCESS\_DENIED.  b. Verify that MemoryOverwriteRequestControl GetVariable() returns a Data value == initial saved value of MemoryOverwriteRequestControl. |
|  |  | MemoryOverwriteRequestControl is locked, the variable must not be erasable(deleted). | 1. Invoke MOR - SetVariable() with valid GUID, variable name, and DataSize == 0.  a. Verify SetVariable() returns EFI\_ACCESS\_DENIED. |

### XX.1.6 MemoryOverwriteRequestControlLock Locked state with key

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 33, 34 | { 0x38a813ac, 0x8eb9, 0x46ce, {0xa8, 0x6b, 0x40, 0x8c, 0x07, 0x5f, 0xc7, 0xed }}, { 0x919b8392, 0xcb78, 0x49ff, {0xa3, 0x18, 0x49, 0x78, 0x76, 0xe0, 0xf8, 0xf8 }} | Changing MemoryOverwriteRequestControlLock to Locked state with a key  must return EFI\_SUCCESS when MemoryOverwriteRequestControlLock is currently unlocked.    MemoryOverwriteRequestControlLock must be locked with key. | 1. Reboot System to make firmware initialize MemoryOverwriteRequestControlLock to Unlocked.  2. Invoke SetVariable() with Valid GUID, variable name, attributes, DataSize, and Data Input = 8-byte user generated key.   1. Verify SetVariable() returns EFI\_SUCCESS. 2. Verify GetVariable() returns Data value 0x02. |
| 35, 36 | { 0x219abaf2, 0x04a9, 0x407c, {0xb5, 0xde, 0xa0, 0x03, 0x6e, 0x65, 0xb0, 0xb9 }}, { 0x9a51640a, 0xff14, 0x402b, {0xb0, 0x57, 0xa0, 0xc4, 0xf7, 0x20, 0x8e, 0x44 }} | Changing MemoryOverwriteRequestControlLock to Unlocked state with the 8 byte key used in the previous SetVariable() Lock with key action  must return EFI\_SUCCESS when MemoryOverwriteRequestControlLock is currently Locked with key.    MemoryOverwriteRequestControlLock must be unlocked. | 1. Invoke SetVariable() with Valid GUID, variable name, attributes, DataSize, and Data Input = 8-byte user generated key that was used to set the lock in test number 33   1. Verify SetVariable() returns EFI\_SUCCESS. 2. Verify GetVariable() returns Data value 0x00. |
| 37, 38, 39 | { 0x8ccd0dbb, 0x9b0a, 0x4bfb, {0xa0, 0x7e, 0xc6, 0x06, 0x8b, 0x91, 0x0d, 0xfb }}, { 0xde6f4e17, 0xe375, 0x4dcb, {0x8f, 0x07, 0x77, 0x7e, 0x62, 0x49, 0xea, 0x2c }},  { 0x4ab6927b, 0x5ee4, 0x4748, {0xa4, 0x9d, 0x2d, 0xf3, 0x70, 0x01, 0x41, 0xd5 }} | Changing MemoryOverwriteRequestControlLock to Unlocked state with an invalid DataSize must return EFI\_ACCESS\_DENIED when MemoryOverwriteRequestControlLock is currently Locked with key.    MemoryOverwriteRequestControlLock must not change. | 1. Invoke SetVariable() with Valid GUID, variable name, attributes, DataSize, and Data Input = 8-byte user generated key.  a. Verify SetVariable() returns EFI\_SUCCESS.  2. Invoke SetVariable() with Valid GUID, variable name, attributes, and DataSize != 8.   1. Verify SetVariable() returns EFI\_ACCESS\_DENIED. 2. Verify GetVariable() returns Data value 0x02. |
| 40, 41 | { 0x33142ecf, 0x0f92, 0x4625, {0xb7, 0xf6, 0x7f, 0x15, 0x25, 0x74, 0xd3, 0x03 }}, { 0xf73c04df, 0x2e42, 0x4174, {0x82, 0x18, 0x0f, 0x25, 0x46, 0x4a, 0x55, 0xe9 }} | Changing MemoryOverwriteRequestControlLock to Unlocked state by passing Data Value 0x00 must return EFI\_ACCESS\_DENIED when MemoryOverwriteRequestControlLock is currently Locked with key.    MemoryOverwriteRequestControlLock must not change. | 1. Invoke SetVariable() with Valid GUID, variable name, attributes, and DataSize, Data = 0x00.   1. Verify SetVariable() returns EFI\_ACCESS\_DENIED. 2. Verify GetVariable() returns Data value 0x02. |
| 42, 43 | { 0xd4eacf82, 0x55d1, 0x4ba1, {0xbe, 0x89, 0x6a, 0x2e, 0x44, 0x0f, 0xc7, 0xc0 }},  { 0xfd88fe63, 0x5ed4, 0x482a, {0x98, 0x44, 0x23, 0x90, 0xf9, 0x09, 0x1c, 0x20 }} | Changing MemoryOverwriteRequestControlLock to Locked without key state must return EFI\_ACCESS\_DENIED when MemoryOverwriteRequestControlLock is currently Locked with key.    MemoryOverwriteRequestControlLock must not change. | 1. Invoke SetVariable() with Valid GUID, variable name, attributes, DataSize = 1, and Data Input = 0x01.   1. Verify SetVariable() returns EFI\_ACCESS\_DENIED. 2. Verify GetVariable() returns Data value 0x02. |
| 44, 45 | { 0x176f16cb, 0xf3a4, 0x4f12, {0x8f, 0x96, 0x4a, 0xc1, 0x8b, 0xdd, 0x6e, 0x3c }}, { 0xf0d5e7b3, 0xdc0f, 0x4a18, {0xb9, 0x78, 0x09, 0x3b, 0x15, 0x55, 0x66, 0xbf }} | Since the MemoryOverwriteRequestControlLock is Set, Changing MemoryOverwriteRequestControl value should return EFI\_ACCESS\_DENIED and the value of MemoryOverwriteRequestControl Should not change | 1. Invoke MOR - GetVariable()with valid GUID, variable name, attributes and DataSize. Save the current value for comparison.  2. Invoke MOR - SetVariable() with valid GUID, variable name, and DataSize. Pass Data value with Bit0 Set High to indicate MORbit0 set.  a. Verify SetVariable() returns EFI\_ACCESS\_DENIED.  b. Verify that MemoryOverwrit-eRequestControl GetVariable() returns a Data value == initial saved value of MemoryOverwriteRequestControl. |
|  |  | MemoryOverwriteRequestControl is locked, the variable must not be erasable(deleted). | 1. Invoke MOR - SetVariable() with valid GUID, variable name, and DataSize == 0.  a. Verify SetVariable() returns EFI\_ACCESS\_DENIED. |
| 46, 47 | { 0x9bf14c4b, 0x2950, 0x4c4e, {0x8b, 0xa3, 0x06, 0x46, 0xf1, 0x27, 0x8c, 0x05 }}, { 0xb0468dee, 0xb1d0, 0x4795, {0x9f, 0xbf, 0xbe, 0x3f, 0x40, 0x2f, 0x3d, 0x6f }} | Changing MemoryOverwriteRequestControlLock to Unlocked state with an 8 byte value that is not equal to the 8-byte key used in the previous SetVariable() Lock with key action  must return EFI\_ACCESS\_DENIED when MemoryOverwriteRequestControlLock is currently Locked with key. The Dictionary Attack Mitigation must also change the Lock value to Locked with no key. | 1. Invoke SetVariable() with Valid GUID, variable name, attributes, DataSize, and Data Input = 8 byte value != 8-byte user generated key from test number 33.   1. Verify SetVariable() returns EFI\_ACCESS\_DENIED. 2. Verify GetVariable() returns Data value 0x01. |
| 48, 49 |  | After the Dictionary Attack Prevention Mechanism is activated, the MorLock should not be able to be unlocked with the correct 8-byte key after the state of the key has already been changed to Locked without key. | 1. SetVariable with Valid GUID, variable name, attributes, DataSize and Data Input = 8 byte key that was used to set the original lock in assertion 37.    1. Verify SetVariable() returns EFI\_ACCESS\_DENIED    2. Verify GetVariable() returns Data Value 0x01. |