Enhanced UEFI Memory Protection for Platform Firmware vendors

Draft Recommendation 0.80

Please provide feedback to uefifb@microsoft.com

4/09/2023

Background

While considerable attention has been devoted to hardware trust anchors and operating system security, attackers have discovered that UEFI firmware is lacking basic memory protection that has been present in other system software for over a decade. Coupled with the inconsistency of security capabilities inherent to vendor firmware implementations, UEFI firmware has become an increasingly attractive system attack vector. Exploiting firmware vulnerabilities can be especially rewarding for an attacker because they can compromise hypervisor-based technologies such as Virtualization Based Security (VBS) and be used to establish boot kits that subvert antivirus and malware detection software in the operating system.

In response to these security concerns, Microsoft is recommending all UEFI platform vendors implement strict memory protections (these may become a requirement in the future). This document will define and clarify what exactly "Memory Protections" are and help to drive a consistent implementation. Given the role of UEFI firmware in the boot process this change will not only impact systems booting Windows but all operating systems, option ROMs, and UEFI pre-boot applications supported on the platform.

A Note on Compatibility

We know this will cause some compatibility challenges, but the more consistent our ecosystem is, the faster it will be to get the extensible parts of our ecosystem aligned. The closing section in this document addresses compatibility mitigations to handle non-compliant modules during this transition period. It will take time for legacy code to be updated to adhere to these new requirements and our collective commitment to progress will pave the way to a more secure and resilient digital future.

Condensed Requirements List

1: The UEFI 2.10 Memory Attribute Protocol must be produced.
2: No address range can be simultaneously readable, writable, and executable.
3: Unallocated memory must be marked EFI_MEMORY_RP.
4: Address space which is not present in the EFI memory map must cause a CPU fault if accessed.
5: Calls to EFI_BOOT_SERVICES.AllocatePages and EFI_BOOT_SERVICES.AllocatePool must return memory with the EFI_MEMORY_XP attribute.
6: Page 0 in physical system memory must be marked EFI_MEMORY_RP.
7: AP and BSP stacks must be marked EFI_MEMORY_XP.
8: AP and BSP stacks must have an EFI_MEMORY_RP page at the bottom to catch overflow.
9: MMIO ranges should be in the EFI memory map and marked EFI_MEMORY_XP.
10: Loaded image sections marked with the data characteristic should be EFI_MEMORY_XP
11: Loaded image sections the code characteristic should be EFI_MEMORY_RO.
12: PE Loaders must check the NX_COMPAT flag of loaded images to determine compatibility with the above memory protection requirements.

Runtime Configurable Protections

To enable memory protection in consumer shipped devices, runtime configurability options need to be present to respond to edge cases and accommodate non-compliant option ROMs. It is up to the platform developer to determine what levers will be available and how faults are handled.

Memory Attribute Protocol

A necessity for increasing the security posture is the availability of the Memory Attribute Protocol. Added in UEFI Spec 2.10, the protocol enables setting and getting EFI memory attributes in the UEFI environment. Project Mu hosts an example implementation of this protocol.

Exception Handling

Increasing the security posture of UEFI implementations will increase the frequency of access violations. Exceptions should either cause a reset or transition the memory protection state into compatibility mode. Platform developers should also take care to ensure their exception handling logic provides enough data to distinguish between fault types and root cause failures. These access violations are often helpful for identifying programmer errors and rooting out critical bugs before they become CVEs.

Memory Management

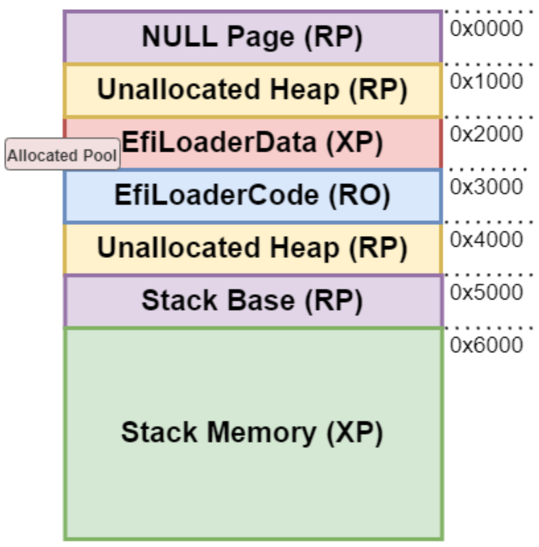
A Note on EFI_MEMORY_RP on ARM Platforms

The definition of EFI_MEMORY_RP is undefined on ARM platforms as of UEFI specification 2.10. However, an implementation of EFI_MEMORY_RP on ARM platforms has been created in EDK2.

General Memory Management

At no point during boot should any addressable memory be readable, writable, and executable. To reach this heightened security bar, all unallocated memory should be marked EFI_MEMORY_RP. Addressable memory ranges which are not present in the EFI memory map should also be read-protected. When a module makes a call to allocate a buffer (even if that buffer is of type EfiBootServicesCode, EfiRuntimeServicesCode, or EfiLoaderCode), the returned page/pool must be non-executable. The module which called for the allocation will be expected to utilize the Memory Attribute Protocol to manipulate the attributes of the buffer to be either writable or executable but not both.

Example Compliant Memory Range



Special Memory Ranges

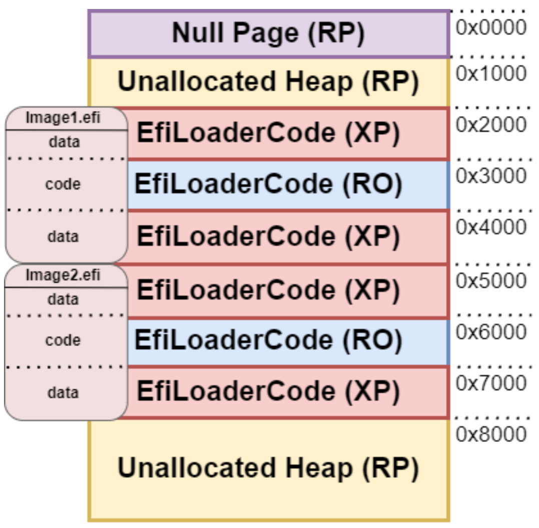
- UEFI must apply EFI_MEMORY_RP to the NULL page to help guard against NULL dereferences.
- AP and BSP stacks must be marked EFI_MEMORY_XP to prevent execution from the stack with a page marked EFI_MEMORY_RP at the base of the stack to prevent stack overflow.
- MMIO ranges should be marked EFI_MEMORY_XP.

PE Loader

On loading and prior to execution of an EFI image, the PE loader must apply EFI_MEMORY_XP to sections marked with the data characteristic and EFI_MEMORY_RO to sections marked with the code characteristic. Applying these page protections requires loaded images to meet the following criteria:

1. Section flags must not combine IMAGE_SCN_MEM_WRITE and IMAGE_SCN_MEM_EXECUTE for any given section.
2. The PE image sections are aligned to page granularity.
3. The PE image must not contain any self-modifying code.

Example of Loaded Image Ranges



NX_COMPAT Characteristic

Many bootloaders and OPROMs will not have implemented support for enhanced protections on image memory, allocated buffers, and other memory ranges. To indicate support for enhanced protections, the PE/COFF IMAGE_DLLCHARACTERISTICS_NX_COMPAT DLL characteristic will be used. Modules with this characteristic are expected to be compliant with enhanced memory protection and should utilize the Memory Attribute Protocol to manipulate the attributes of memory they allocate. If a module is loaded without this characteristic, the platform should enter compatibility mode.

Compatibility Mode

To provide a more consistent and predictable environment across UEFI implementations, we are providing a definition for compatibility mode here. Errant modules and unexpected faults blocking boot should enter compatibility mode which triggers the following deviations from the enhanced memory protection definition:

1. All new memory allocated will be readable, writable, and executable.
2. All images loaded from the start of compatibility mode will no longer have restrictive access attributes applied to the memory ranges in which they are loaded.
3. The Memory Attribute Protocol will be uninstalled.
4. Page zero will be mapped.
5. Legacy BIOS memory (the lower 1MB range) will be mapped as readable, writable, and executable.

A Note on User Notification of Compatibility Mode

When a system is in compatibility mode, it is important that the user receives some notification during boot which could take the form of a dialogue box, color bar, or other visual indicator. At the start of this journey toward more strict memory protections, we believe how compatibility mode is visualized should be left up to the platform. In the future, platform developers should consider adding a TPM (Trusted Platform Module) measurement when the system enters compatibility mode to have platform enforcement of memory protections.

Closing

It will take substantial work to update legacy code to adhere to these new security standards. These protection mitigations offer significant value to end users by heightening both real and perceived security, and to OEMs by reducing the number of issues in the future. As we push toward updating incompatible modules, OEMs should apply strict protections during product development which meet or exceed the standard outlined in this document to help catch programmer error and reduce the volume of unsafe firmware code entering the ecosystem.

Microsoft is committed to this effort and is prepared to work with partners as we move toward a more secure UEFI ecosystem.

Additional References

PE Format: PE Format - Win32 apps | Microsoft Learn

Requirements for 3rd party signing: UEFI CA Memory Mitigation Requirements for Signing - Windows drivers | Microsoft Learn

UEFI Test Tool: mu_plus/UefiTestingPkg/AuditTests/PagingAudit at release/202311 · microsoft/mu_plus (github.com)