UEFI

By Contributors to Wikimedia projects

Published: 2004-07-29 · Archived: 2026-04-05 16:32:07 UTC

Unified Extensible Firmware Interface	
Abbreviation	UEFI
Status	Published
Year started	2006[a]
Latest version	2.11[1] 16 December 2024
Organization	UEFI Forum
Related standards	• ACPI • UEFI Platform Initialization
Predecessor	BIOS on IBM PC compatible computers[b]
Domain	Firmware
Website	uefi.org

Boot order selection menu on a Lenovo ThinkPad T470 with both UEFI and BIOS support

The UEFI implementation is usually stored on NOR-based flash memory[2][3][4] located on the motherboard. Various I/O protocols can be used, SPI being the most common.

Unified Extensible Firmware Interface (**UEFI**, [*citation needed*] as an initial)[c] is a specification for the firmware architecture of a computing platform. When a computer is powered on, the UEFI implementation is typically the first that runs, before starting the operating system. Examples include AMI Aptio, Phoenix SecureCore, TianoCore EDK II, and InsydeH2O.

UEFI replaces the BIOS that was present in the boot ROM of all personal computers that are IBM PC compatible,[5][6] although it can provide backwards compatibility with the BIOS using CSM booting. Unlike its predecessor, BIOS, which is a de facto standard originally created by IBM as proprietary software, UEFI is an open standard maintained by an industry consortium. Like BIOS, most UEFI implementations are proprietary.

Intel developed the original *Extensible Firmware Interface* (*EFI*) specification. The last Intel version of EFI was 1.10 released in 2005. Subsequent versions have been developed as UEFI by the UEFI Forum.

UEFI is independent of platform and programming language, but C is used for the reference implementation TianoCore EDKII.

The original motivation for EFI came during early development of the first Intel–HP Itanium systems in the mid-1990s. BIOS limitations had become too restrictive for the larger server platforms Itanium was targeting for.[7] The effort to address these concerns began in 1998 and was initially called *Intel Boot Initiative*.[8] It was later renamed to *Extensible Firmware Interface* (EFI).[9][10]

The first open-source UEFI implementation, Tiano, was released by Intel in 2004. Tiano has since then been superseded by EDK[11] and EDK II[12] and is now maintained by the TianoCore community.[13]

In July 2008, Intel ceased its development of the EFI specification at version 1.10 and contributed it to the Unified EFI Forum, which has developed the specification as the *Unified Extensible Firmware Interface* (UEFI). The original EFI specification remains owned by Intel, which exclusively provides licenses for EFI-based products, but the UEFI specification is owned by the UEFI Forum.[7][14]

Version 2.0 of the UEFI specification was released on 31 January 2006. It added cryptography and security[*vague*].

Version 2.1 of the UEFI specification was released on 7 January 2007. It added network authentication and the user interface architecture ("Human Interface Infrastructure" in UEFI).

Version 2.3.1 of the UEFI specification was released on 6 April 2011. It added Secure Boot, as well as ARM architecture support.

In October 2018, Arm announced Arm ServerReady, a compliance certification program for landing the generic off-the-shelf operating systems and hypervisors on Arm-based servers. The program requires the system firmware to comply with Server Base Boot Requirements (SBBR). SBBR requires UEFI, ACPI and SMBIOS compliance. In October 2020, Arm announced the extension of the program to the edge and IoT market. The new program name is Arm SystemReady. Arm SystemReady defined the Base Boot Requirements (BBR) specification that currently provides three recipes, two of which are related to UEFI: 1) SBBR: which requires UEFI, ACPI and SMBIOS compliance suitable for enterprise-level operating environments such as Windows, Red Hat Enterprise Linux, and VMware ESXi; and 2) EBBR: which requires compliance to a set of UEFI interfaces as defined in the Embedded Base Boot Requirements (EBBR) suitable for embedded environments such as Yocto. Many Linux and BSD distributions can support both recipes.

In December 2018, Microsoft announced Project Mu, a fork of TianoCore EDK II used in Microsoft Surface and Hyper-V products. The project promotes the idea of firmware as a service.[15]

The latest UEFI specification, version 2.11, was published in December 2024.[16]

Processor compatibility

[edit]

UEFI supports processor architectures that are 32-bit or higher. However, only processors with a little-endian mode are supported.[16]:section 1.9.1 The UEFI specification, version 2.11, has official documentation for the following processor architectures:[16]:section 3.5.1.1

- x86 (IA-32, x86-64)
- Itanium (IA-64)
- ARM (AArch32, AArch64)
- RISC-V (32-bit, 64-bit, 128-bit)
- LoongArch (32-bit, 64-bit)

Unofficial UEFI support is under development for POWERPC64 by implementing TianoCore EDK II on top of OPAL,[17] the OpenPOWER abstraction layer, running in little-endian mode.[18] For MIPS, there also exists an unofficial project, based on the original EDK.[19][20] However, both projects have since been abandoned as of November 2016 and September 2015 respectively.

UEFI only allows executing UEFI applications that match the firmware's bit-width, even if the processor supports smaller or larger bit-widths. For example, a 64-bit UEFI firmware may only execute 64-bit UEFI applications,

even if the processor has a 32-bit processor mode.[16]:sections 2.3.2 and 2.3.4 Some low-end computers have been shipped with 32-bit UEFI firmware running on 64-bit CPUs.[21] Once a UEFI application ends the boot services and gets granted full control over the system, it becomes possible to change the processor execution mode.[16]:sections 2.3.2 and 2.3.4 However, calling runtime services requires shortly changing back to the original processor mode,[22] as runtime services may only be called from the same processor mode as the firmware implementation.[16]:sections 2.3.2 and 2.3.4

The Linux kernel added support for booting 64-bit kernels on 32-bit UEFI firmware implementations with x86-64 CPUs since version 3.15, requiring the UEFI boot loader to support the EFI handover protocol.[23] The EFI handover protocol allows UEFI boot loaders to defer the UEFI initialization to the kernel's EFI boot stub, so that only the kernel does the UEFI initialization.[24][25][26][*needs update*]

Disk device compatibility

[edit]

In addition to the standard PC disk partition scheme that uses a master boot record (MBR), UEFI also works with the GUID Partition Table (GPT) partitioning scheme, which is free from many of the limitations of MBR. In particular, the MBR limits on the number and size of disk partitions (up to four primary partitions per disk, and up to 2 TB (2×2^{40} bytes) per disk) are relaxed. More specifically, GPT allows for a maximum disk and partition size of 8 ZiB (8×2^{70} bytes) with 512 byte sectors.[27] The UEFI specification only supports FAT12/16/32[16]:section 13.3 partitions that are on GPT or MBR disks as well as El Torito-formatted optical discs.[16]:section 13.3.2 Although GPT is a part of the UEFI standard, it may also be usable by BIOS PCs to boot an operating system off of.[27][28]

Support for GPT in Linux is enabled by turning on the option `CONFIG_EFI_PARTITION` (EFI GUID Partition Support) during kernel configuration.[29] This option allows Linux to recognize and use GPT disks after the system firmware passes control over the system to Linux.

For reverse compatibility, Linux can use GPT disks in BIOS-based systems for both data storage and booting, as both GRUB 2 and Linux are GPT-aware. Such a setup is usually referred to as *BIOS-GPT*.[*citation needed*] As GPT incorporates the protective MBR, a BIOS-based computer can boot from a GPT disk using a GPT-aware boot loader stored in the protective MBR's bootstrap code area.[27] In the case of GRUB, such a configuration requires a BIOS boot partition for GRUB to embed its second-stage code due to absence of the post-MBR gap in GPT partitioned disks (which is taken over by the GPT's *Primary Header* and *Primary Partition Table*). Commonly 1 MB in size, this partition's Globally Unique Identifier (GUID) in GPT scheme is 21686148-6449-6E6F-744E-656564454649 and is used by GRUB only in BIOS-GPT setups. From GRUB's perspective, no such partition type exists in case of MBR partitioning. This partition is not required if the system is UEFI-based because no embedding of the second-stage code is needed in that case.[28][27]

UEFI systems can access GPT disks and boot directly from them, which allows Linux to use UEFI boot methods. Booting Linux from GPT disks on UEFI systems involves creation of an EFI system partition (ESP), which contains UEFI applications such as bootloaders, operating system kernels, and utility software.[30][31] Such a

setup is usually referred to as *UEFI-GPT*, while ESP is recommended to be at least 512 MB in size and formatted with a FAT32 filesystem for maximum compatibility.[27]

For backward compatibility, some UEFI implementations also support booting from MBR-partitioned disks through the Compatibility Support Module (CSM) that provides legacy BIOS compatibility.[*citation needed*] In that case, booting Linux on UEFI systems is the same as on legacy BIOS-based systems.

Some of the EFI's practices and data formats mirror those of Microsoft Windows.[*further explanation needed*][32][33]

Windows 11, 64-bit versions of Windows Vista SP1/SP2 and 7, and both 32-bit and 64-bit versions of Windows 8, 8.1, and 10 can boot from a GPT disk that is larger than 2 TB.

EFI defines two types of services: *boot services* and *runtime services*. Boot services are available only while the firmware owns the platform (i.e., before the `ExitBootServices()` call), and they include text and graphical consoles on various devices, and bus, block and file services. Runtime services are still accessible while the operating system is running; they include services such as date, time and NVRAM access.

Graphics Output Protocol (GOP) services
: The *Graphics Output Protocol* (GOP) provides runtime services; see also Graphics features section below. The operating system is permitted to directly write to the framebuffer and bit blit provided by GOP during runtime mode.[34]

UEFI memory map services

SMM services

ACPI services

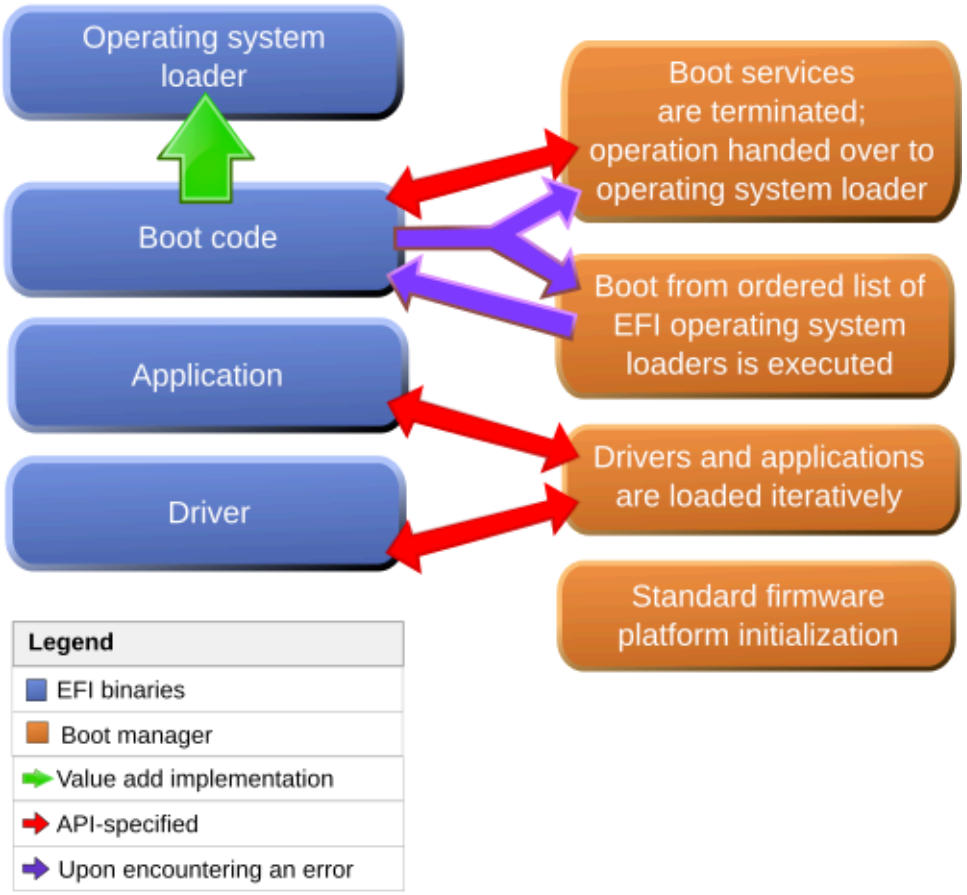
SMBIOS services

Devicetree services (for RISC processors)

Variable services
: UEFI variables provide a way to store data, in particular non-volatile data. Some UEFI variables are shared between platform firmware and operating systems. Variable namespaces are identified by GUIDs, and variables are key/value pairs. For example, UEFI variables can be used to keep crash messages in NVRAM after a crash for the operating system to retrieve after a reboot.[35]

Time services
: UEFI provides time services. Time services include support for time zone and daylight saving fields, which allow the hardware real-time clock to be set to local time or UTC.[16]:section 8.3 On machines using a PC-AT real-time clock, by default the hardware clock still has to be set to local time for compatibility with BIOS-based Windows,[33] unless using recent versions and an entry in the Windows registry is set to indicate the use of UTC.

Interaction between the EFI boot manager and EFI drivers



Beyond loading an OS, UEFI can run *UEFI applications*, which reside as files on the EFI system partition. They can be executed from the UEFI Shell, by the firmware's boot manager, or by other UEFI applications. *UEFI applications* can be developed and installed independently of the original equipment manufacturers (OEMs).

A type of UEFI application is an OS boot loader such as GRUB, rEFInd, systemd-boot, and Windows Boot Manager, which loads some OS files into memory and executes them. Also, an OS boot loader can provide a user interface to allow the selection of another UEFI application to run. Utilities like the UEFI Shell are also UEFI applications.

EFI defines protocols as a set of software interfaces used for communication between two binary modules. All EFI drivers must provide services to others via protocols. The EFI Protocols are similar to the BIOS interrupt calls.

In addition to standard instruction set architecture-specific device drivers, EFI provides for a ISA-independent device driver stored in non-volatile memory as *EFI byte code* or *EBC*. System firmware has an interpreter for EBC images. In that sense, EBC is analogous to Open Firmware, the ISA-independent firmware used in PowerPC-based Apple Macintosh and Sun Microsystems SPARC computers, among others.

Some architecture-specific (non-EFI Byte Code) EFI drivers for some device types can have interfaces for use by the OS. This allows the OS to rely on EFI for drivers to perform basic graphics and network functions before, and if, operating-system-specific drivers are loaded.

In other cases, the EFI driver can be filesystem drivers that allow for booting from other types of disk volumes. Examples include *efifs* for 37 file systems (based on GRUB2 code),[36] used by Rufus for chain-loading NTFS

ESPs.[37]

The EFI 1.0 specification defined a UGA (Universal Graphic Adapter) protocol as a way to support graphics features. UEFI did not include UGA and replaced it with GOP (Graphics Output Protocol).[38]

UEFI 2.1 defined a "Human Interface Infrastructure" (HII) to manage user input, localized strings, fonts, and forms (in the HTML sense). These enable original equipment manufacturers (OEMs) or independent BIOS vendors (IBVs) to design graphical interfaces for pre-boot configuration. UEFI uses UTF-16 to encode strings by default; since at least UEFI 2.4, it allows use ASCII to encode English-only strings.

Most early UEFI firmware implementations were console-based. Today many UEFI firmware implementations are GUI-based.[*citation needed*]

EFI system partition

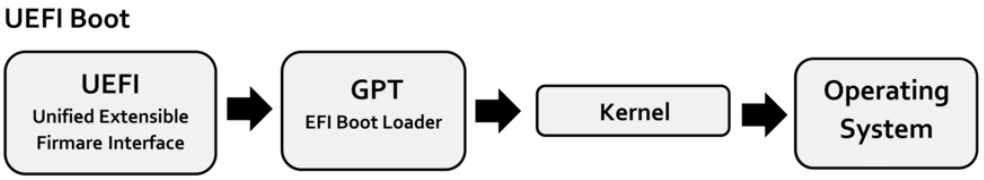
[edit]

An EFI system partition, often abbreviated to ESP, is a data storage device partition that is used in computers adhering to the UEFI specification. Accessed by the UEFI firmware when a computer is powered up, it stores UEFI applications and the files these applications need to run, including operating system boot loaders. Supported partition table schemes include MBR and GPT, as well as El Torito volumes on optical discs.[16]:section 2.6.2 For use on ESPs, UEFI defines a specific version of the FAT file system, which is maintained as part of the UEFI specification and independently from the original FAT specification, encompassing the FAT32, FAT16 and FAT12 file systems.[16]:section 13.3[39] The ESP also provides space for a boot sector as part of the backward BIOS compatibility.[*citation needed*]

Computers are started up by a process which has been called booting: the computer loads its operating software by means of a very small program built into the hardware, which typically loads a program, still small, to load and start the operating system (OS).

Unlike the legacy PC BIOS, UEFI does not rely on boot sectors on the computer's data storage, defining instead a boot manager as part of the UEFI specification. When a computer is powered on, the boot manager checks the boot configuration and, based on its settings, then executes the specified OS boot loader or operating system kernel. The boot configuration is defined by variables stored in the computer's persistent NVRAM storage, including variables that indicate the file system paths to OS loaders or OS kernels.

OS boot loaders can be automatically detected by UEFI, which enables easy booting from removable devices such as USB flash drives. This automated detection relies on standardized file paths to the OS boot loader, with the path varying depending on the computer architecture. The format of the file path is defined as <EFI_SYSTEM_PARTITION>\EFI\BOOT\BOOT<MACHINE_TYPE_SHORT_NAME>.EFI; for example, the file path to the OS loader on an x86-64 system is \efi\boot\bootx64.efi,[16]:section 3.5.1.1 and \efi\boot\bootaa64.efi on ARM64 architecture.



Boot process

Booting UEFI systems from GPT-partitioned disks is commonly called *UEFI-GPT booting*, although the UEFI specification requires MBR partition tables to be fully supported.[16]:section 13.3.2 Some UEFI firmware implementations immediately switch to BIOS-based CSM booting depending on the type of the boot disk's partition table, effectively preventing UEFI booting from an EFI System Partition on MBR-partitioned disks; [*citation needed*] such a boot scheme is commonly called *UEFI-MBR*.

It is also common for a boot manager to have a textual user interface to allow the user to select the desired OS (or setup utility) from a list of available boot options.

On PC platforms, the BIOS firmware that supports UEFI boot can be called *UEFI BIOS*, although it may not support CSM boot method, as modern x86 PCs deprecate use of CSM.

To ensure backward compatibility, UEFI firmware implementations on PC-class machines could support booting in legacy BIOS mode from MBR-partitioned disks through the *Compatibility Support Module (CSM)* that provides legacy BIOS compatibility. In this scenario, booting is performed in the same way as on legacy BIOS-based systems, by ignoring the partition table and relying on the content of a boot sector.[*citation needed*]

BIOS-style booting from MBR-partitioned disks is commonly called *BIOS-MBR*, regardless of it being performed on UEFI or legacy BIOS-based systems. Furthermore, booting legacy BIOS-based systems from GPT disks is also possible, and such a boot scheme is commonly called *BIOS-GPT*.

The *Compatibility Support Module* allows legacy operating systems and some legacy option ROMs that do not support UEFI to still be used.[40] It also provides required legacy System Management Mode (SMM) functionality, called *CompatibilitySmm*, as an addition to features provided by the UEFI SMM. An example of such a legacy SMM functionality is providing USB legacy support for keyboard and mouse, by emulating their classic PS/2 counterparts.[40]

In November 2017, Intel announced that it planned to phase out support CSM for client platforms by 2020.[41]

In July 2022, Kaspersky Labs published information regarding a Rootkit designed to chain boot malicious code on machines using Intel's H81 chipset and the Compatibility Support module of affected motherboards.[42]

In August 2023, Intel announced that it planned to phase out CSM support for server platforms by 2024.[43]

Currently [*when?*] most computers based on Intel platforms do not support CSM.[*citation needed*]

The UEFI specification includes support for booting over network via the Preboot Execution Environment (PXE). PXE booting network protocols include Internet Protocol (IPv4 and IPv6), User Datagram Protocol (UDP),

Dynamic Host Configuration Protocol (DHCP), Trivial File Transfer Protocol (TFTP) and iSCSI.[16]:924–1509[44]

OS images can be remotely stored on storage area networks (SANs), with Internet Small Computer System Interface (iSCSI) and Fibre Channel over Ethernet (FCoE) as supported protocols for accessing the SANs.[16] [*page needed*][45][46]

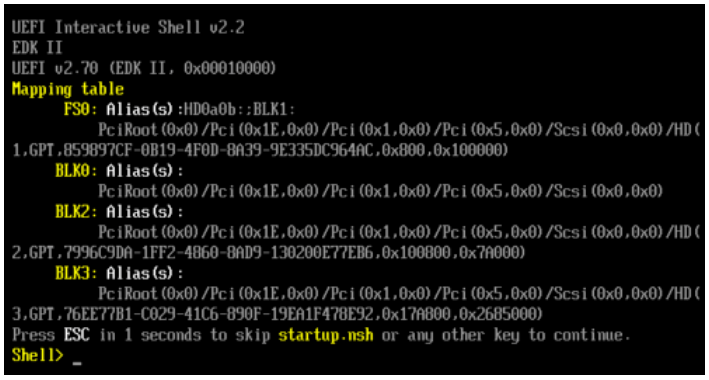
Version 2.5 of the UEFI specification adds support for accessing boot images over HTTP.[47]

See also: Secure Boot criticism

Example of an active Secure Boot as detected by rEFInd boot manager

The UEFI specification defines a protocol known as *Secure Boot*, which can secure the boot process by preventing the loading of UEFI drivers or OS boot loaders that are not signed with an acceptable digital signature. When Secure Boot is enabled, it is initially placed in "setup" mode, which allows a public key known as the "platform key" (PK) to be written to the firmware. Once the key is written, Secure Boot enters "User" mode, where only UEFI drivers and OS boot loaders signed with the platform key can be loaded by the firmware. Additional "key exchange keys" (KEK) can be added to a database stored in memory to allow other certificates to be used, but they must still have a connection to the private portion of the platform key.[48] Secure Boot can also be placed in "Custom" mode, where additional public keys can be added to the system that do not match the private key.[49]

Secure Boot is supported by Windows 8 and 8.1, Windows Server 2012 and 2012 R2, Windows 10, Windows Server 2016, 2019, and 2022, and Windows 11, VMware vSphere 6.5[50] and a number of Linux distributions including Fedora (since version 18), openSUSE (since version 12.3), RHEL (since version 7), CentOS (since version 7[51]), Debian (since version 10),[52] Ubuntu (since version 12.04.2), Linux Mint (since version 21.3).,[53] [54] and AlmaLinux OS (since version 8.4[55]). As of January 2025, FreeBSD support is in a planning stage.[56]



Example of a UEFI shell 2.2 session

UEFI provides a shell environment, which can be used to execute other UEFI applications, including UEFI boot loaders.[*citation needed*] Apart from that, commands available in the UEFI shell can be used for obtaining various other information about the system or the firmware, including getting the memory map (`memmap`), modifying boot manager variables (`bcfg`), running partitioning programs (`diskpart`), loading UEFI drivers, and editing text files (`edit`).[57][58]

Source code for a UEFI shell can be downloaded from the Intel's TianoCore UDK/EDK2 project.[59] A pre-built ShellBinPkg is also available.[60] Shell v2 works best in UEFI 2.3+ systems and is recommended over Shell v1 in those systems. Shell v1 should work in all UEFI systems.[61][62]

Methods used for launching UEFI shell depend on the manufacturer and model of the system motherboard. Some of them already provide a direct option in firmware setup for launching, e.g. compiled x86-64 version of the shell needs to be made available as `<EFI_SYSTEM_PARTITION>/SHELLX64.EFI` . Some other systems have an already embedded UEFI shell which can be launched by appropriate key press combinations.[63] For other systems, the solution is either creating an appropriate USB flash drive or adding manually (`bcfg`) a boot option associated with the compiled version of shell.[58]

The following is a list of commands supported by the EFI shell.[57]

- alias
- attrib
- bcfg
- cd
- cls
- comp
- cp
- date
- dblk
- dh
- dmpstore
- echo
- Edd30
- EddDebug

- edit
- err
- guid
- help
- load
- ls
- map
- mem
- memmap
- mkdir
- mm
- mode
- mount
- pause
- pci
- reset
- rm
- set
- stall
- time
- type
- unload
- ver
- vol

Extensions to UEFI can be loaded from virtually any non-volatile storage device attached to the computer. For example, an original equipment manufacturer (OEM) can distribute systems with an EFI system partition on the hard drive, which would add additional functions to the standard UEFI firmware stored on the motherboard's ROM.

UEFI Capsule defines a Firmware-to-OS firmware update interface, marketed as modern and secure.[64] Windows 8, Windows 8.1, Windows 10,[65] and Fwupd for Linux each support the UEFI Capsule.

Like BIOS, UEFI initializes and tests system hardware components, and then loads the boot loader from a mass storage device or through a network connection. In x86 systems, the UEFI firmware is usually stored in the NOR flash chip of the motherboard, and the boot process is more complex, for example PCI Express devices detection and initialization.[66][67] In some ARM-based Android and Windows Phone devices, the UEFI boot loader is stored in the eMMC or eUFS flash memory.

UEFI machines can have one of the following classes, which were used to help ease the transition to UEFI:[68]

- Class 0: Legacy BIOS
- Class 1: UEFI with a CSM interface and no external UEFI interface. The only UEFI interfaces are internal to the firmware.

- Class 2: UEFI with CSM and external UEFI interfaces, eg. UEFI Boot.
- Class 3: UEFI without a CSM interface and with an external UEFI interface.
- Class 3+: UEFI class 3 that has Secure Boot enabled.[69]

Starting from the 10th Gen Intel Core, Intel no longer provides Legacy Video BIOS for the iGPU (Intel Graphics Technology). Legacy boot with those CPUs requires a Legacy Video BIOS, which can still be provided by a video card.[*citation needed*]

SEC – Security Phase

[edit]

This is the first stage of the UEFI boot but may have platform specific binary code that precedes it. (e.g., Intel ME, AMD PSP, CPU microcode). It consists of minimal code written in assembly language for the specific architecture. It initializes a temporary memory (often CPU cache-as-RAM (CAR), or SoC on-chip boot processor) and serves as the system's software root of trust with the option of verifying PEI before hand-off.

- Initialization of temporary memory for next stage, PEI.
- Root of trust, by the means of verifying the integrity of PEI.
- Passing handoff information to the PEI foundation. The information includes the location and size of temporary memory, location and size of stack and state of the platform.

PEI – Pre-EFI Initialization

[edit]

The second stage of UEFI boot consists of a dependency-aware dispatcher that loads and runs PEI modules (PEIMs) to handle early hardware initialization tasks such as main memory initialization (initialize memory controller and DRAM) and firmware recovery operations. Additionally, it is responsible for discovery of the current boot mode and handling many ACPI S3 operations. In the case of ACPI S3 resume, it is responsible for restoring many hardware registers to a pre-sleep state. PEI also uses CAR. Initialization at this stage involves creating data structures in memory and establishing default values within these structures.[4]

This stage has several components including PEI foundation, PEIMs and PPI. Due to less resources available in this stage, this stage must be minimal and do minimal preparations for the next stage, DXE, Which is more richer.

After SEC phase hand off, platform responsibility is taken by PEI Foundation. Its responsibilities are:

- Successful dispatching of PEIMs (pre-EFI initialization modules).
- Initializing permanent memory (RAM).
- Handing over to the next stage, DXE.
- Facilitating the communication of PEIMs called PPI.

This component is responsible for invoking PEIMs and managing their dependencies.

Pre-EFI Initialization Modules

[edit]

These are minimal PEI drivers that are responsible for initialization of hardware components, such as permanent memory, CPU, chipset and motherboard. Each of the PEIMs have single responsibilities and focus on single initialization. These drivers come from different vendors.

PEIMs-to-PEIMs Interfaces

[edit]

This is a data structure that composed of GUID pairs of pointers. PPIs are discovered by PEIMs through PEI services.

After minimal initialization of the system for DXE, PEI foundation locates and passes control to DXE. The PEI foundation dispatches DXE foundation through special PPI called IPL(Initial Program Load).

DXE – Driver Execution Environment

[edit]

This stage consists of C modules and a dependency-aware dispatcher. With main memory now available, CPU, chipset, mainboard and other I/O devices are initialized in DXE and BDS. Initialization at this stage involves assigning EFI device paths to the hardware connected to the motherboard, and transferring configuration data to the hardware.[4]

BDS – Boot Device Select (Boot Manager)

[edit]

BDS is a part of the DXE.[70][71] In this stage, boot devices are initialized, UEFI drivers or Option ROMs of PCI devices are executed according to architecturally defined variables called **NVRAM**.

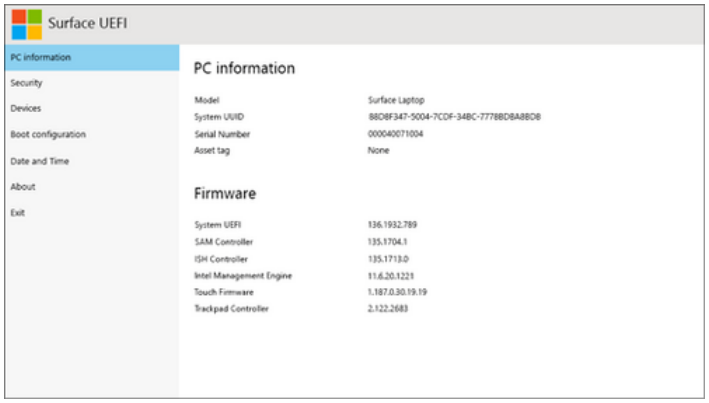
TSL – Transient System Load

[edit]

This is the stage between boot device selection and hand-off to the OS. At this point one may enter a UEFI shell, or execute a UEFI application such as the OS boot loader.

The UEFI hands off to the operating system (OS) after ExitBootServices() is executed. A UEFI compatible OS is now responsible for exiting boot services triggering the firmware to unload all no longer needed code and data, leaving only runtime services code/data, e.g. SMM and ACPI.[72][*failed verification*] A typical modern OS will prefer to use its own programs (such as kernel drivers) to control hardware devices.

When a legacy OS is used, CSM will handle this call ensuring the system is compatible with legacy BIOS expectations.



Microsoft Surface UEFI, the UEFI used on all Surface models made after 2015

Intel's implementation of EFI is the *Intel Platform Innovation Framework*, codenamed *Tiano*. Tiano runs on Intel's XScale, Itanium, IA-32 and x86-64 processors, and is proprietary software, although a portion of the code has been released under the BSD license or Eclipse Public License (EPL) as TianoCore EDK II. TianoCore can be used as a payload for coreboot.[73]

Phoenix Technologies' implementation of UEFI is branded as SecureCore Technology (SCT).[74] American Megatrends offers its own UEFI firmware implementation known as Aptio,[75] while Insyde Software offers InsydeH2O,[76] and Byosoft offers ByoCore.

In December 2018, Microsoft released an open source version of its TianoCore EDK2-based UEFI implementation from the Surface line, Project Mu.[77]

An implementation of the UEFI API was introduced into the Universal Boot Loader (Das U-Boot) in 2017.[78] On the ARMv8 architecture Linux distributions use the U-Boot UEFI implementation in conjunction with GNU GRUB for booting (e.g. SUSE Linux[79]), the same holds true for OpenBSD.[80] For booting from iSCSI iPXE can be used as a UEFI application loaded by U-Boot.[81]

Intel's first Itanium workstations and servers, released in 2000, implemented EFI 1.02.

Hewlett-Packard's first Itanium 2 systems, released in 2002, implemented EFI 1.10. These systems were able to boot Windows, Linux, FreeBSD and HP-UX. OpenVMS added UEFI capability in June 2003.

In January 2006, Apple Inc. shipped its first Intel-based Macintosh computers. These systems used EFI instead of Open Firmware, which had been used on its previous PowerPC-based systems.[82] On 5 April 2006, Apple first released Boot Camp, which produces a Windows drivers disk and a non-destructive partitioning tool to allow the installation of Windows XP or Vista without requiring a reinstallation of Mac OS X (now macOS). A firmware update was also released that added BIOS compatibility to its EFI implementation. Subsequent Macintosh models shipped with the newer firmware.[83]

During 2005, more than one million Intel systems shipped with Intel's implementation of UEFI.[84][*failed verification*] New mobile, desktop and server products, using Intel's implementation of UEFI, started shipping in 2006. For instance, boards that use the Intel 945 chipset series use Intel's UEFI firmware implementation.

Since 2005, EFI has also been implemented on non-PC architectures, such as embedded systems based on XScale cores.[84]

The EDK (EFI Developer Kit) includes an NT32 target, which allows EFI firmware and EFI applications to run within a Windows application. However, no direct hardware access is allowed by EDK NT32. This means only a subset of EFI application and drivers can be executed by the EDK NT32 target.

In 2008, more x86-64 systems adopted UEFI. While many of these systems still allow booting only the BIOS-based OSes via the Compatibility Support Module (CSM) (thus not appearing to the user to be UEFI-based), other systems started to allow booting UEFI-based OSes. For example, IBM x3450 server, MSI motherboards with ClickBIOS and HP EliteBook Notebook PCs.

In 2009, IBM shipped System x machines (x3550 M2, x3650 M2, iDataPlex dx360 M2) and BladeCenter HS22 with UEFI capability. Dell shipped PowerEdge T610, R610, R710, M610 and M710 servers with UEFI capability. More commercially available systems are mentioned in a UEFI whitepaper.[85]

In 2011, major vendors (such as ASRock, Asus, Gigabyte, and MSI) launched several consumer-oriented motherboards using the Intel 6-series LGA 1155 chipset and AMD 9 Series AM3+ chipsets with UEFI.[86]

With the release of Windows 8 in October 2012, Microsoft's certification requirements now require that computers include firmware that implements the UEFI specification. Furthermore, if the computer supports the "Connected Standby" feature of Windows 8 (which allows devices to have power management comparable to smartphones, with an almost instantaneous return from standby mode), then the firmware is not permitted to contain a Compatibility Support Module (CSM). As such, systems that support Connected Standby are incapable of booting Legacy BIOS operating systems.[87][88]

In October 2017, Intel announced that it would remove legacy PC BIOS support from all its products by 2020, in favor of UEFI Class 3.[89] By 2019, all computers based on Intel platforms no longer have legacy PC BIOS support.

A operating system that can be booted from (U)EFI is called a (U)EFI-aware operating system, defined by (U)EFI specification. Here the term *booted from a (U)EFI* means directly booting the system using a (U)EFI operating system loader stored on any storage device. The default location for the operating system loader is `<EFI_SYSTEM_PARTITION>/BOOT/BOOT<MACHINE_TYPE_SHORT_NAME>.EFI`, where short name of the machine type can be `IA32`, `X64`, `IA64`, `ARM` or `AA64`.[16]:section 3.5.1.1 Some operating systems vendors may have their own boot loaders. They may also change the default boot location.

- The Linux kernel has been able to use EFI at boot time since early 2000s,[90] using the elilo EFI boot loader and, more recently, EFI versions of GRUB[91] or systemd-boot. Grub+Linux also supports booting from a GUID partition table without UEFI.[28] The distribution Ubuntu added support for UEFI Secure Boot as of version 12.10.[92] Furthermore, the Linux kernel can be compiled with the option to run as an EFI bootloader on its own through the EFI boot stub feature. In Linux kernel, either ACPI protocol (usually adapt on PC type devices) or DeviceTree protocol (usually adapt on smartphone and tablet type devices) can be used with UEFI.[93]

- HP-UX has used (U)EFI as its boot mechanism on IA-64 systems since 2002.
- OpenVMS has used EFI on IA-64 since its initial evaluation release in December 2003, and for production releases since January 2005.[94] OpenVMS on x86-64 also uses UEFI to boot the operating system.[95]
- Apple uses EFI for its line of Intel-based Macs. Mac OS X v10.4 Tiger and Mac OS X v10.5 Leopard implements EFI v1.10 in 32-bit mode even on 64-bit CPUs, but full support arrived with OS X v10.8 Mountain Lion.[96][*failed verification*]
- The Itanium versions of Windows 2000 (Advanced Server Limited Edition and Datacenter Server Limited Edition; based on the pre-release Windows Server 2003 codebase) implemented EFI 1.10 in 2002. Windows XP 64-bit Edition, Windows 2000 Advanced Server Limited Edition (pre-release Windows Server 2003) and Windows Server 2003 for IA-64, all of which are for the Intel Itanium family of processors, implement EFI, a requirement of the platform through the DIG64 specification.[97]
- Microsoft introduced UEFI for x64 Windows operating systems with Windows Vista SP1[98] and Windows Server 2008 however only UGA (Universal Graphic Adapter) 1.1 or Legacy BIOS INT 10h is supported; Graphics Output Protocol (GOP) is not supported. Therefore, PCs running 64-bit versions of Windows Vista SP1, Windows Vista SP2, Windows 7, Windows Server 2008 and Windows Server 2008 R2 are compatible with UEFI Class 2.[99][100] 32-bit UEFI was originally not supported since vendors did not have any interest in producing native 32-bit UEFI firmware because of the mainstream status of 64-bit computing.[101] Windows 8 finally introduced further optimizations for UEFI systems, including Graphics Output Protocol (GOP) support,[102] a faster startup, 32-bit UEFI support, and Secure Boot support.[103] [104] Since Windows 8, the UEFI firmware with ACPI protocol is a mandatory requirement for ARM-based Microsoft Windows operating systems. Microsoft began requiring UEFI to run Windows with Windows 11, [105] with IoT Enterprise editions of Windows 11 since version 24H2 exempt from the requirement.[106]
- On 5 March 2013, the FreeBSD Foundation awarded a grant to a developer seeking to add UEFI support to the FreeBSD kernel and bootloader.[107] The changes were initially stored in a discrete branch of the FreeBSD source code, but were merged into the mainline source on 4 April 2014 (revision 264095); the changes include support in the installer as well.[108] UEFI boot support for amd64 first appeared in FreeBSD 10.1 and for arm64 in FreeBSD 11.0.[109]
- NetBSD supports UEFI since version 8.0[110] for i386 and amd64 architectures as well as on various ARM platforms since version 9.0[111].
- Oracle Solaris 11.1 and later support UEFI boot for x86 systems with UEFI firmware version 2.1 or later. GRUB 2 is used as the boot loader on x86.[112]
- OpenBSD 5.9[113] introduced UEFI boot support for 64-bit x86 systems using its own custom loader, OpenBSD 6.0 extended that support to include ARMv7.[114]
- illumos added basic UEFI support in October 2017.[115]
- ArcaOS supports UEFI booting since the 5.1 release.[116] ArcaOS' UEFI support emulates specific BIOS functionality which the operating system depends on (particularly interrupts INT 10H and INT 13H).[117] [118]

With virtualization

[edit]

- HP Integrity Virtual Machines provides UEFI boot on HP Integrity Servers. It also provides a virtualized UEFI environment for the guest UEFI-aware OSes.
- Intel hosts an Open Virtual Machine Firmware project on SourceForge.[119]
- VMware Fusion 3 software for Mac OS X can boot Mac OS X Server virtual machines using UEFI.
- VMware Workstation prior to version 11 unofficially supports UEFI, but is manually enabled by editing the .vmx file.[120] VMware Workstation version 11 and above supports UEFI, independently of whether the physical host system is UEFI-based. VMware Workstation 14 (and accordingly, Fusion 10) adds support for the Secure Boot feature of UEFI.[121][122]
- The VMware ESXi 5.0 hypervisor officially supports UEFI. Version 6.5 adds support for Secure Boot.[123][124]
- VirtualBox has implemented UEFI since 3.1,[125] but is limited to Unix/Linux operating systems and Windows 8 and later (does not work with Windows Vista x64 and Windows 7 x64).[126][127]
- QEMU/KVM can be used with the Open Virtual Machine Firmware (OVMF) provided by TianoCore.[128]
- The second generation of the Microsoft Hyper-V virtual machine supports virtualized UEFI.[129]
- Google Cloud Platform Shielded VMs support virtualized UEFI to enable Secure Boot.[130]

UEFI implementation flaws have been exploited to gain persistence, the ability to maintain malicious access to a compromised system even after system reboot, operating system reinstallation, and even partial physical part replacement, such as of corrupted PCI persistent flash storage. There were vulnerabilities in 2023 even with Secure Boot enabled.[131] In 2023 Microsoft published a warning about BlackLotus UEFI malware.[132]

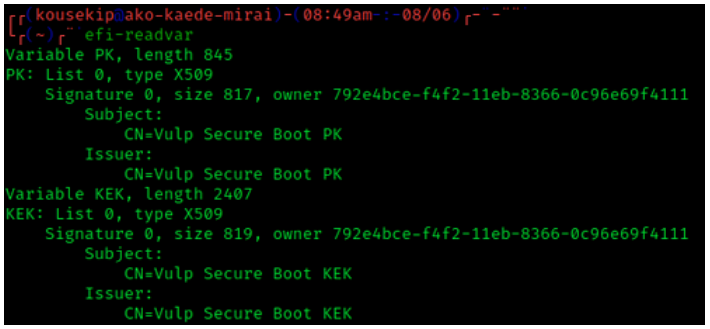
Applications development

[edit]

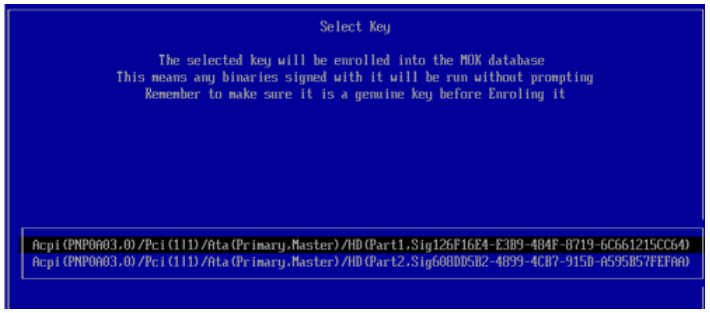
EDK2 Application Development Kit (EADK) makes it possible to use standard C library functions in UEFI applications. EADK can be freely downloaded from the Intel's TianoCore UDK / EDK2 SourceForge project. As an example, a port of the Python interpreter is made available as a UEFI application by using the EADK.[133] The development has moved to GitHub since .UDK2015.[134]

Numerous digital rights activists have protested UEFI. Ronald G. Minnich, a co-author of coreboot, and Cory Doctorow, a digital rights activist, have criticized UEFI as an attempt to remove the ability of the user to truly control the computer.[135][136] It does not solve the BIOS's long-standing problems of requiring two different drivers—one for the firmware and one for the operating system—for most hardware.[137]

Open-source project TianoCore also provides UEFIs.[138] TianoCore lacks the specialized firmware drivers and modules that initialize chipset functions, but TianoCore is one of many payload options of coreboot. The development of coreboot requires cooperation from chipset manufacturers to provide the specifications needed to develop initialization drivers.



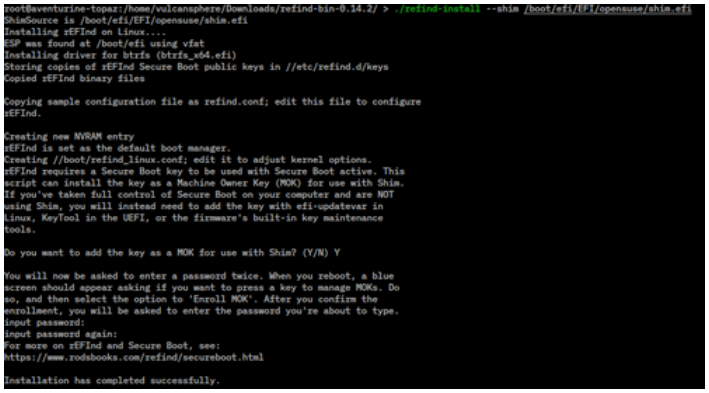
Examples of custom Secure Boot public keys



MokManager, a part of shim bootloader used to enroll Machine Owner Key (MOK) to UEFI system

In 2011, Microsoft announced that computers certified to run its Windows 8 operating system had to ship with Microsoft's public key enrolled and Secure Boot enabled, which implies that using UEFI is a requirement for these devices.[139][140] Following the announcement, the company was accused by critics and free-software/open-source advocates (including the Free Software Foundation) of trying to use the Secure Boot functionality of UEFI to hinder or outright prevent the installation of alternative operating systems such as Linux. Microsoft denied that the Secure Boot requirement was intended to serve as a form of lock-in and clarified its requirements by stating that x86-based systems certified for Windows 8 must allow Secure Boot to enter custom mode or be disabled, but not on systems using the ARM architecture.[49][141] Windows 10 allows OEMs to decide whether or not Secure Boot can be managed by users of their x86 systems.[142]

Other developers raised concerns about the legal and practical issues of implementing support for Secure Boot on Linux systems in general. Former Red Hat developer Matthew Garrett noted that conditions in the GNU General Public License version 3 may prevent the use of the GNU GRand Unified Bootloader without a distribution's developer disclosing the private key (however, the Free Software Foundation has since clarified its position, assuring that the responsibility to make keys available was held by the hardware manufacturer),[143][92] and that it would also be difficult for advanced users to build custom kernels that could function with Secure Boot enabled without self-signing them.[141] Other developers suggested that signed builds of Linux with another key could be provided, but noted that it would be difficult to persuade OEMs to ship their computers with the required key alongside the Microsoft key.[6]



Screenshot of installation of rEFInd boot manager, using shim and Machine Owner Key (MOK) for Secure Boot support

Several major Linux distributions have developed different implementations for Secure Boot. Garrett himself developed a minimal bootloader known as a shim, which is a precompiled, signed bootloader that allows the user to individually trust keys provided by Linux distributions.[144] Ubuntu 12.10 uses an older version of shim[*which?*] pre-configured for use with Canonical's own key that verifies only the bootloader and allows unsigned kernels to be loaded; developers believed that the practice of signing only the bootloader is more feasible, since a trusted kernel is effective at securing only the user space, and not the pre-boot state for which Secure Boot is designed to add protection. That also allows users to build their own kernels and use custom kernel modules as well, without the need to reconfigure the system.[92][145][146] Canonical also maintains its own private key to sign installations of Ubuntu pre-loaded on certified OEM computers that run the operating system, and also plans to enforce a Secure Boot requirement as well—requiring both a Canonical key and a Microsoft key (for compatibility reasons) to be included in their firmware. Fedora also uses shim,[*which?*] but requires that both the kernel and its modules be signed as well.[145] shim has Machine Owner Key (MOK) that can be used to sign locally compiled kernels and other software not signed by distribution maintainer, without changing the Secure Boot mode to setup mode.[147][148]

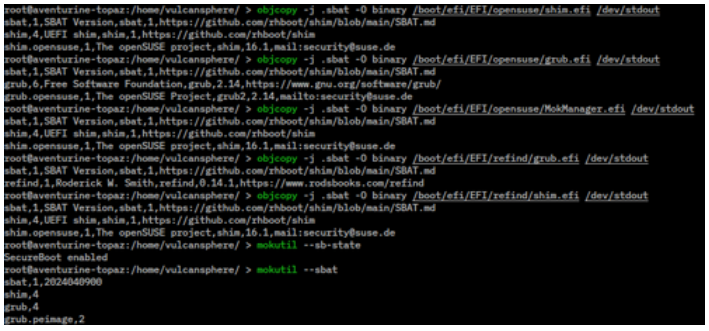
It has been disputed whether the operating-system kernel and its modules must be signed as well; while the UEFI specifications do not require it, Microsoft has asserted that their contractual requirements do, and that it reserves the right to revoke any certificates used to sign code that can be used to compromise the security of the system.[146] In Windows, if Secure Boot is enabled, all kernel drivers must be digitally signed; non-WHQL drivers may be refused to load. In February 2013, another Red Hat developer attempted to submit a patch to the Linux kernel that would allow it to parse Microsoft's authenticode signing using a master X.509 key embedded in PE files signed by Microsoft. However, the proposal was criticized by Linux creator Linus Torvalds, who attacked Red Hat for supporting Microsoft's control over the Secure Boot infrastructure.[149]

On 26 March 2013, the Spanish free-software development group Hispalinux filed a formal complaint with the European Commission, contending that Microsoft's Secure Boot requirements on OEM systems were "obstructive" and anti-competitive.[150]

At the Black Hat conference in August 2013, a group of security researchers presented a series of exploits in specific vendor implementations of UEFI that could be used to exploit Secure Boot.[151]

In August 2016 it was reported that two security researchers had found the "golden key" security key Microsoft uses in signing operating systems.[152] Technically, no key was exposed, however, an exploitable binary signed by the key was. This allows any software to run as though it was genuinely signed by Microsoft and exposes the possibility of rootkit and bootkit attacks. This also makes patching the fault impossible, since any patch can be replaced (downgraded) by the (signed) exploitable binary. Microsoft responded in a statement that the vulnerability only exists in ARM architecture and Windows RT devices, and has released two patches; however, the patches do not (and cannot) remove the vulnerability, which would require key replacements in end-user firmware to fix.[*citation needed*]

On 1 March 2023, researchers from ESET Cybersecurity Firm reported "The first in-the-wild UEFI bootkit bypassing UEFI Secure Boot" named "BlackLotus" in their public analyses findings describing the theory behind its mechanics exploiting the patches that "do not (and cannot) remove the vulnerability".[153][154]



Example of Secure Boot Advanced Targeting (SBAT) metadata in UEFI applications

In August 2024, the Windows 11 and Windows 10 security updates applied the Secure Boot Advanced Targeting (SBAT) settings to device's UEFI NVRAM, which caused some Linux distributions to fail to load. SBAT is a protocol that supported in new versions of Windows Boot Manager and shim, which refuse buggy or vulnerable intermediate bootloaders (usually older versions of Windows Boot Manager and GRUB) to load in the boot process. The change was reverted the next month.[155]

In June 2025, LWN.net reported that the Microsoft UEFI CA 2011 certificate will be expired in June 2026, which may refuse some Linux to load if Secure Boot is enabled.[156] However, in TianoCore EDK II as well as many commercial UEFI implementations (such as AMI Aptio), time/date check for Secure Boot certificate is usually disabled by default.[157]

Many Linux distributions support UEFI Secure Boot as of January 2025, such as RHEL (RHEL 7 and later), CentOS (CentOS 7 and later[158]), Ubuntu, Fedora, Debian (Debian 10 and later[159]), OpenSUSE, and SUSE Linux Enterprise.[160]

The increased prominence of UEFI firmware in devices has also led to a number of technical problems blamed on their respective implementations.[161]

Following the release of Windows 8 in October 2012, it was discovered that certain Lenovo computer models with Secure Boot had UEFI that was checking for the presence of "Windows Boot Manager" or "Red Hat Enterprise Linux" in the descriptive string of an UEFI-supported operating system, otherwise refusing to load it.[162] Other

problems were encountered by several Toshiba laptop models with Secure Boot that were missing certain certificates required for its proper operation.[161]

In January 2013 a bug was publicized regarding the UEFI implementation on some Samsung laptops, which caused them to be bricked after installing a Linux distribution in UEFI mode. While potential conflicts with a kernel module designed to access system features on Samsung laptops were initially blamed (also prompting kernel maintainers to disable the module on UEFI systems as a safety measure), Matthew Garrett discovered that the bug was actually triggered by storing too many UEFI variables to memory, and that the bug could also be triggered under Windows under certain conditions. He determined that the offending kernel module had caused kernel message dumps to be written to the firmware, thus triggering the bug.[35][163][164]

- ACPI – Computer firmware interface standard
- Bootloader – Software responsible for starting the computer
- MoonBounce – UEFI malware
- OpenBIOS – Free software implementation of Open Firmware
- System Management BIOS (SMBIOS)
- Trusted Platform Module (TPM)
- UEFI Platform Initialization (UEFI PI)
- UEFITool – Software program

1. ^ Originally started in 1998 as Intel Boot Initiative and later as Extensible Firmware Interface (EFI), which was deprecated in 2005 and replaced by UEFI.
2. ^ Part of the BIOS that is required for booting an operating system that is not UEFI-compatible can be implemented as a CSM DXE module, see § CSM booting.
3. ^ Historically also written as Unified EFI, when UEFI was the newly introduced successor to EFI.

1. ^ *"UEFI Forum Releases the UEFI 2.11 Specification and the PI 1.9 Specification To Streamline User Implementations | Unified Extensible Firmware Interface Forum"*. *uefi.org. UEFI Forum. Retrieved 22 December 2024.*
2. ^ *"Documentation - Winbond"*.
3. ^ *"Microsoft Surface Laptop 7 (13.8-inch) Chip ID"*. *22 June 2024.*
4. ^ Jump up to: **a** **b** **c** *Zimmer, Vincent; Rothman, Michael; Marisetty, Suresh (2017). Beyond BIOS: Developing with the Unified Extensible Firmware Interface, Third Edition. Walter de Gruyter GmbH & Co KG. ISBN 978-1-5015-0569-0.*
5. ^ *Kinney, Michael (1 September 2000). "Solving BIOS Boot Issues with EFI" (PDF). pp. 47–50. Archived from the original (PDF) on 23 January 2007. Retrieved 14 September 2010.*
6. ^ Jump up to: **a** **b** *"MS denies secure boot will exclude Linux". The Register. 23 September 2011. Retrieved 24 September 2011.*
7. ^ Jump up to: **a** **b** *"Emulex UEFI Implementation Delivers Industry-leading Features for IBM Systems" (PDF). Emulex. Retrieved 14 September 2010.*
8. ^ *Extensible Firmware Interface (EFI) and Unified EFI (UEFI), Intel, archived from the original on 5 January 2010*
9. ^ *Wei, Dong (2006), "foreword", Beyond BIOS, Intel Press, ISBN 978-0-9743649-0-2*

10. ^ *"1.10 Specification overview"*, *Extensible Firmware Interface, Intel*
11. ^ *"Git mirror of EDK"*. *GitHub*. *19 March 2019.*
12. ^ *"EDK II"*. *GitHub*. *8 August 2019.*
13. ^ *"What is TianoCore?"*.
14. ^ *About*, *Unified EFI Forum, "Q: What is the relationship between EFI and UEFI? A: The UEFI specification is based on the EFI 1.10 specification published by Intel with corrections and changes managed by the Unified EFI Forum. Intel still holds the copyright on the EFI 1.10 specification, but has contributed it to the Forum so that the Forum can evolve it. There will be no future versions of the EFI specification, but customers who license it can still use it under the terms of their license from Intel. The license to the Unified EFI Specification comes from the Forum, not from Intel"*
15. ^ *"Microsoft announces Project Mu, an open-source release of the UEFI core"*. *20 December 2018.*
16. ^ Jump up to: ***a*** ***b*** ***c*** ***d*** ***e*** ***f*** ***g*** ***h*** ***i*** ***j*** ***k*** ***l*** ***m*** ***n*** ***o*** ***p*** *Unified Extensible Firmware Interface (UEFI) Specification (PDF). Uefi.org (Technical report). 2.11. UEFI Forum. 21 November 2024. Retrieved 5 January 2026.*
17. ^ *"GitHub - andreiw/ppc64le-edk2: TianoCore UEFI for OPAL/PowerNV (PPC64/PowerPC64 Little-Endian)". GitHub. 3 May 2021.*
18. ^ *"Tianocore for OpenPOWER". Firmware Security. 12 October 2015.*
19. ^ *kontais (3 September 2015). "EFI-MIPS". SourceForge.*
20. ^ *"GitHub - ncroxon/gnu-efi:Develop EFI applications for ARM-64, ARM-32, x86_64, IA-64 (IPF), IA-32 (x86), and MIPS platforms using the GNU toolchain and the EFI development environment". GitHub.*
21. ^ *James Brian Richardson (22 July 2015). "Why Cheap Systems Run 32-bit UEFI on x64 Systems". Archived from the original on 24 December 2020.*
22. ^ *Fleming, Matt (4 March 2014). "x86/efi: Add early thunk code to go from 64-bit to 32-bit - kernel/git/torvalds/linux.git - Linux kernel source tree". Linux kernel mailing list.*
23. ^ *"Linux kernel 3.15, Section 1.3. EFI 64-bit kernels can be booted from 32-bit firmware". kernelnewbies.org. 8 June 2014. Retrieved 15 June 2014.*
24. ^ *"x86, efi: Handover Protocol". LWN.net. 19 July 2012. Retrieved 15 June 2014.*
25. ^ *"Linux kernel documentation: Documentation/efi-stub.txt". kernel.org. 1 February 2014. Retrieved 15 June 2014.*
26. ^ *"1. The Linux/x86 Boot Protocol - The Linux Kernel documentation". kernel.org.*
27. ^ Jump up to: ***a*** ***b*** ***c*** ***d*** ***e*** *Smith, Roderick W. (3 July 2012). "Make the most of large drives with GPT and Linux". IBM. Archived from the original on 9 July 2012. Retrieved 25 September 2013.*
28. ^ Jump up to: ***a*** ***b*** ***c*** *"Installation". 3.4 BIOS installation. GNU GRUB. Retrieved 25 September 2013.*
29. ^ *"block/partitions/Kconfig (3.11.1)". CONFIG_EFI_PARTITION (line #247). kernel.org. Retrieved 25 September 2013.*
30. ^ *"GRUB and the boot process on UEFI-based x86 systems". redhat.com. Retrieved 14 November 2013.*
31. ^ *"UEFI Booting 64-bit Redhat Enterprise Linux 6". fpmurphy.com. September 2010. Archived from the original on 12 July 2013. Retrieved 14 November 2013.*
32. ^ IBM PC Real Time Clock should run in UT. Cl.cam.ac.uk. Retrieved on 30 October 2013.
33. ^ Jump up to: ***a*** ***b*** *Garrett, Matthew (19 January 2012). "EFI and Linux: The Future Is Here, and It's Awful". linux.conf.au 2012. Archived from the original on 13 November 2021. Retrieved 2 April 2012.*
34. ^ *"What is efifb? — The Linux Kernel documentation". www.kernel.org. Retrieved 24 November 2020.*

35. ^ Jump up to: *a* *b* *"Samsung UEFI bug: Notebook bricked from Windows"*. *The H. Retrieved 27 February 2013.*
36. ^ *"Free Software EFI Drivers"*.
37. ^ *Batard, Pete (13 March 2020). "pbatard/uefi-ntfs". GitHub.*
38. ^ *"Intel Embedded Graphics Drivers FAQ: BIOS and firmware"*. *Intel. Retrieved 19 May 2014.*
39. ^ *"Technical Note TN2166: Secrets of the GPT"*. *developer.apple.com. 6 November 2006. Retrieved 6 May 2015.*
40. ^ Jump up to: *a* *b* *"Intel Platform Innovation Framework for EFI" (PDF). Compatibility Support Module Specification (revision 0.97). Intel. 4 September 2007. Retrieved 6 October 2013.*
41. ^ *"The PC BIOS will be killed off by 2020 as Intel plans move to pure UEFI"*. *Ars Technica. Retrieved 29 May 2018.*
42. ^ *"CosmicStrand: the discovery of a sophisticated UEFI firmware rootkit"*. *Securelist by Kaspersky. Retrieved 4 August 2022.*
43. ^ *"Removal of Legacy Boot Support for Intel Platforms Technical Advisory"*. *Intel.*
44. ^ *"Red Hat Enterprise Linux 6 Installation Guide"*. *30.2.2. Configuring PXE boot for EFI. Red Hat. Retrieved 9 October 2013.*
45. ^ *El-Haj-Mahmoud, Samer (July 2013). "Advances in Pre-OS Networking in UEFI 2.4" (PDF). Hewlett-Packard. Retrieved 29 May 2019.*
46. ^ *Racherla, Sangam; Erdenberger, Silvio; Rajagopal, Harish; Ruth, Kai (January 2014). Storage and Network Convergence Using FCoE and iSCSI (PDF) (2nd ed.). IBM Redbooks. Retrieved 20 April 2022.*
47. ^ *"New UEFI HTTP Boot support in UEFI 2.5"*. *firmwaresecurity.com. 9 May 2015. Retrieved 13 August 2015.*
48. ^ *Edge, Jake (15 June 2011). "UEFI and "secure boot""*. *LWN.net. Retrieved 9 September 2012.*
49. ^ Jump up to: *a* *b* *"Windows 8 Secure Boot: The Controversy Continues"*. *PC World. Retrieved 9 September 2012.*
50. ^ *"Secure Boot for ESXi 6.5 - Hypervisor Assurance"*. *VMware vSphere Blog. 4 May 2017. Retrieved 18 August 2017.*
51. ^ HowTos/UEFI - CentOS Wiki
52. ^ *Larabel, Michael (30 April 2018). "Debian Making Progress on UEFI SecureBoot Support in 2018". Phoronix. Phoronix Media. Retrieved 23 May 2018.*
53. ^ *Garrett, Matthew (27 December 2012). "Secure Boot distribution support". Mjg59.dreamwidth.org. Retrieved 20 March 2014.*
54. ^ *"Linux Mint Secure boot". Linux Mint. Retrieved 12 January 2024.*
55. ^ *"8.4 | AlmaLinux Wiki". wiki.almalinux.org. Retrieved 10 April 2024.*
56. ^ *"SecureBoot". FreeBSD Wiki. FreeBSD. Retrieved 16 June 2015.*
57. ^ Jump up to: *a* *b* *"EFI Shells and Scripting"*. *Intel. Archived from the original on 16 July 2019. Retrieved 25 September 2013.*
58. ^ Jump up to: *a* *b* *"UEFI Shell Specification Version 2.0, Errata A" (PDF). Unified EFI. May 2012. Retrieved 25 September 2013.*
59. ^ *"EDK2: ShellPkg"*. *GitHub. Retrieved 18 March 2020.*
60. ^ *"tianocore/edk2: releases"*. *GitHub.*

61. ^ *"Email Archive: edk2-devel". [edk2] Inclusion of UEFI shell in Linux distro iso. SourceForge. 2012. Archived from the original on 3 October 2013. Retrieved 25 September 2013.*
62. ^ *"TianoCore on SourceForge". Shell FAQ. Intel. Retrieved 25 September 2013.*
63. ^ *"Basic Instructions for Using EFI for Server Configuration on Intel Server Boards and Intel Server Systems" (PDF). Intel. 2008. Retrieved 25 September 2013.*
64. ^ *"Signed Capsule Update". tianocore-docs.github.io.*
65. ^ *barrygolden. "Windows UEFI firmware update platform - Windows drivers". docs.microsoft.com. Retrieved 25 September 2020.*
66. ^ *"Getting Started | Microchip Technology". www.microchip.com. Retrieved 24 December 2020.*
67. ^ *"Framework-Laptop-16/Mainboard/Mainboard_Interfaces_Schematic_Framework_Laptop_16_7040_Series.PDF at main · FrameworkComputer/Framework-Laptop-16" (PDF). GitHub.*
68. ^ *Barry, Peter; Crowley, Patrick (2012). Modern embedded computing : designing connected, pervasive, media-rich systems. Waltham, MA: Morgan Kaufmann Publishers. p. 169. ISBN 978-0-12-394407-8. OCLC 810455404.*
69. ^ *"Intel schrapt bios-compatibiliteit uefi in 2020". Tweakers (in Dutch). Retrieved 30 December 2022.*
70. ^ *"PI Boot Flow · tianocore/Tianocore.github.io Wiki". GitHub.*
71. ^ *"Engineering Services" (PDF).*
72. ^ *"The Unified Extensible Firmware Interface (UEFI)". The Linux Kernel documentation.*
73. ^ *"TianoCore - coreboot". Archived from the original on 22 October 2020. Retrieved 25 May 2012.*
74. ^ *"SecureCore Tiano™". Phoenix Technologies. Archived from the original on 6 September 2010. Retrieved 14 September 2010.*
75. ^ *"Aptio: The Complete UEFI Product Solution" (PDF). American Megatrends. Retrieved 2 May 2018.*
76. ^ *"Why US?". Insyde Software Corp. Archived from the original on 27 December 2018. Retrieved 2 May 2018.*
77. ^ *"Microsoft Announces "Project Mu" For Open-Source UEFI Alternative To TianoCore". Phoronix. Retrieved 20 December 2018.*
78. ^ *"Marrying U-Boot UEFI and GRUB" (PDF). Retrieved 12 September 2018.*
79. ^ *"UEFI on Top of U-Boot" (PDF). Archived from the original (PDF) on 11 September 2018. Retrieved 12 September 2018.*
80. ^ *"Installing OpenBSD 6.3 on Raspberry 3". Archived from the original on 21 November 2018. Retrieved 12 September 2018.*
81. ^ *"iSCSI booting with U-Boot and iPXE". Archived from the original on 31 July 2020. Retrieved 18 May 2020.*
82. ^ Apple Computer. "Universal Binary Programming Guidelines, Second Edition: Extensible Firmware Interface (EFI) Archived 24 July 2008 at the Wayback Machine"
83. ^ Apple's Transition from Open Firmware to Extensible Firmware Interface, mactech, 2007.
84. ^ Jump up to: ***a*** ***b*** *"Intel Platform Innovation Framework for UEFI Overview". Intel. Retrieved 14 September 2010.*
85. ^ *Evaluating UEFI using Commercially Available Platforms and Solutions (PDF), UEFI, May 2011, archived from the original (PDF) on 22 March 2012*
86. ^ Asus P67 Motherboard Preview.

87. ^ *"Windows Hardware Certification Requirements for Client and Server Systems". Microsoft. January 2013. "System.Fundamentals.Firmware.CS.UEFISecureBoot.ConnectedStandby ... Platforms shall be UEFI Class Three (see UEFI Industry Group, Evaluating UEFI using Commercially Available Platforms and Solutions, version 0.3, for a definition) with no Compatibility Support Module installed or installable. BIOS emulation and legacy PC/AT boot must be disabled."*
88. ^ *"Microsoft: All You Need to Know About Windows 8 on ARM". PC Magazine. Archived from the original on 6 September 2013. Retrieved 30 September 2013.*
89. ^ *Richardson, Brian (30 October 2017). ""Last Mile" Barriers to Removing Legacy BIOS" (PDF). Retrieved 22 November 2017.*
90. ^ Announcement of release 3.5pre1 by maintainer Brett Johnson made on 27 February 2004.
91. ^ *EFI version of Grub, Debian GNU/Linux, retrieved 1 May 2008*
92. ^ Jump up to: ***a*** ***b*** ***c*** *"Ubuntu will use GRUB 2 for its Secure Boot implementation". The H Online. Retrieved 28 October 2012.*
93. ^ *"The Unified Extensible Firmware Interface (UEFI) — The Linux Kernel documentation". www.kernel.org. Retrieved 22 March 2026.*
94. ^ *OpenVMS Release History, HP, archived from the original on 5 January 2009, retrieved 16 September 2008*
95. ^ *"State of the Port to x86-64" (PDF). vmssoftware.com. 6 October 2017. Archived from the original (PDF) on 22 September 2020. Retrieved 9 September 2020.*
96. ^ *rEFIt — Windows Vista and EFI, SourceForge, archived from the original on 5 September 2008, retrieved 31 May 2008*
97. ^ *"Extensible Firmware Interface", Windows Server TechCenter, Microsoft, archived from the original on 30 August 2006*
98. ^ Unified Extensible Firmware Interface (UEFI) Implementation Guidelines
99. ^ *Ersek, Laszlo (January 2015). "Open Virtual Machine Firmware (OVMF) Status Report". Linux KVM Project. Retrieved 13 November 2022.*
100. ^ Boot to UEFI Mode or legacy BIOS mode
101. ^ *"Unified Extended Firmware Interface support in Windows Vista". Microsoft. 26 October 2006. Archived from the original on 27 July 2020. Retrieved 12 June 2010. "Microsoft determined that vendors would not have any interest in producing native UEFI 32-bit firmware because of the current status of mainstream 64-bit computing and platform costs. Therefore, Microsoft originally did not to ship support for 32-bit UEFI implementations."*
102. ^ Microsoft Basic Display Driver
103. ^ *"Microsoft Touts Incredible Windows 8 Boot Times". Retrieved 9 September 2011.*
104. ^ *Brodkin, Jon (21 September 2011). "Windows 8 secure boot could complicate Linux installs". Ars Technica. Retrieved 23 September 2011.*
105. ^ *"Find Windows 11 specs, features, and computer requirements". Microsoft.*
106. ^ *"Minimum System Requirements for Windows IoT Enterprise". Microsoft Learn. 22 May 2024. Retrieved 7 June 2024.*
107. ^ *"FreeBSD to get UEFI support". The H. Retrieved 7 March 2013.*
108. ^ *"UEFI - FreeBSD Wiki". FreeBSD.org. Retrieved 19 June 2014.*
109. ^ *"uefi(8)". www.freebsd.org. Retrieved 11 January 2021.*

110. ^ *"Announcing NetBSD 8.0 (July 17, 2018)". www.netbsd.org. Retrieved 8 February 2026.*
111. ^ *"Announcing NetBSD 9.0 (Feb 14, 2020)". www.netbsd.org. Retrieved 8 February 2026.*
112. ^ *"Oracle Solaris 11.1 — What's New" (PDF). oracle.com. Retrieved 4 November 2013.*
113. ^ *"OpenBSD 5.9". www.openbsd.org. Retrieved 11 September 2016.*
114. ^ *"OpenBSD 6.0". www.openbsd.org. Retrieved 11 September 2016.*
115. ^ *"8422 uts: basic UEFI support for illumos". www.illumos.org. Retrieved 17 December 2024.*
116. ^ *Proven, Liam (4 September 2023). "ArcaOS 5.1 gives vintage OS/2 a UEFI facelift for the 21st century". The Register. Retrieved 4 September 2023.*
117. ^ *"Booting ArcaOS on UEFI hardware (demonstration)". youtube.com. 8 August 2019. Retrieved 22 September 2020.*
118. ^ *Sanders, James (13 August 2019). "Modern OS/2 distribution ArcaOS adds support for booting via UEFI". techrepublic.com. Archived from the original on 21 October 2019. Retrieved 4 September 2023.*
119. ^ *Open Virtual Machine Firmware, SourceForge, archived from the original on 6 October 2011*
120. ^ *"VMWare Workstation EFI firmware | VMware Communities". Communities.vmware.com. 3 October 2012. Retrieved 28 February 2014.*
121. ^ *"Using EFI/UEFI firmware in a VMware Virtual Machine | VMware Communities". Communities.vmware.com. 6 December 2014. Retrieved 18 January 2016.*
122. ^ *"Announcing VMware Workstation 14 - VMware Workstation Zealot". VMware Workstation Zealot. 22 August 2017. Retrieved 2 August 2018.*
123. ^ *"What's New in vSphere 5.0". Vmware.com. Archived from the original on 23 January 2014. Retrieved 28 February 2014.*
124. ^ *"VMware vSphere 6.5 Release Notes". pubs.vmware.com. Archived from the original on 13 January 2017. Retrieved 13 January 2017.*
125. ^ *3.1 Changelog, VirtualBox, archived from the original on 28 September 2010*
126. ^ *Ticket 7702, VirtualBox*
127. ^ *"Statement by sr. software engineer at Oracle", Forum, VirtualBox*
128. ^ *"Testing secureboot with KVM". FedoraProject. Retrieved 28 February 2014.*
129. ^ *"What's New in Hyper-V for Windows Server 2012 R2". MicrosoftTechNet. Retrieved 24 June 2013.*
130. ^ *"Shielded VMs". Retrieved 16 February 2019.*
131. ^ *Spring, Jonathan; Radesky, Sandra (3 August 2023). "A Call to Action: Bolster UEFI Cybersecurity Now". Cybersecurity and Infrastructure Security Agency (CISA).*
132. ^ *"Microsoft Releases Guidance for the BlackLotus Campaign". Cybersecurity and Infrastructure Security Agency (CISA). 11 April 2023.*
133. ^ *"TianoCore on SourceForge: EDK2 Application Development Kit (EADK)". Intel. Retrieved 25 September 2013.*
134. ^ *"Tianocore: UDK". GitHub.*
135. ^ *"Interview: Ronald G Minnich". Fosdem. 6 February 2007. Retrieved 14 September 2010.*
136. ^ *Doctorow, Cory (27 December 2011), The Coming War on General Purpose Computation, retrieved 25 September 2013*
137. ^ *"coreboot (aka LinuxBIOS): The Free/Open-Source x86 Firmware". YouTube. 31 October 2008. Retrieved 14 September 2010.*
138. ^ *"Welcome", TianoCore, SourceForge, archived from the original on 23 April 2012*

139. ^ *"Next-gen boot spec could forever lock Linux off Windows 8 PCS"*.
140. ^ *"Windows 8 secure boot could complicate Linux installs"*. *21 September 2011*.
141. ^ Jump up to: **a** **b** *"Is Microsoft Blocking Linux Booting on ARM Hardware?"*. *Computerworld UK. Archived from the original on 22 December 2014. Retrieved 6 March 2012.*
142. ^ *"Windows 10 to make the Secure Boot alt-OS lock out a reality"*. *Ars Technica. 20 March 2015. Retrieved 21 March 2015.*
143. ^ *"Free Software Foundation recommendations for free operating system distributions considering Secure Boot"*. *Free Software Foundation. Retrieved 18 March 2020.*
144. ^ *"Shimming your way to Linux on Windows 8 PCs"*. *ZDNet. Retrieved 26 February 2013.*
145. ^ Jump up to: **a** **b** *Willis, Nathan (27 June 2012). "Ubuntu details its UEFI Secure Boot plans". Linux Weekly News. Retrieved 11 September 2012.*
146. ^ Jump up to: **a** **b** *"No Microsoft certificate support in Linux kernel says Torvalds"*. *The H. Retrieved 26 February 2013.*
147. ^ *Smith, Roderick W. (4 November 2012). "Managing EFI Boot Loaders for Linux: Dealing with Secure Boot (Using the Shim Program)". Roderick W. Smith's Web Page. Retrieved 17 January 2025.*
148. ^ *Edge, Jake (17 October 2012). "Another approach to UEFI secure boot". LWN.net. Retrieved 7 February 2026.*
149. ^ *"Linus Torvalds: I will not change Linux to "deep-throat Microsoft""*. *Ars Technica. 26 February 2013. Retrieved 26 February 2013.*
150. ^ *"Exclusive: Open software group files complaint against Microsoft to EU"*. *Reuters. 26 March 2013. Retrieved 26 March 2013.*
151. ^ *"Researchers demo exploits that bypass Windows 8 Secure Boot"*. *IT World. Archived from the original on 5 August 2013. Retrieved 5 August 2013.*
152. ^ *Mendelsohn, Tom (12 August 2016). "Secure Boot snafu: Microsoft leaks backdoor key, firmware flung wide open [Updated]". Ars Technica. Retrieved 12 August 2016.*
153. ^ *Smolár, Martin (1 March 2023). "BlackLotus UEFI bootkit: Myth confirmed". welivesecurity.com. Retrieved 1 March 2023.*
154. ^ *Goodin, Dan (6 March 2023). "Stealthy UEFI malware bypassing Secure Boot enabled by unpatchable Windows flaw". Ars Technica. Retrieved 6 March 2023.*
155. ^ *WindowsCommunications (26 August 2024). "Windows 11, version 23H2 known issues and notifications". learn.microsoft.com. Retrieved 3 September 2024.*
156. ^ *Edge, Jake (16 July 2025). "Linux and Secure Boot certificate expiration". LWN.net. Retrieved 12 September 2025.*
157. ^ *"Windows Secure Boot certificate expiration and CA updates - Microsoft Support"*. *support.microsoft.com. Retrieved 12 September 2025.*
158. ^ *"HowTos/UEFI"*. *CentOS Wiki. Retrieved 10 November 2020.*
159. ^ *"SecureBoot"*. *Debian Wiki. Retrieved 10 November 2020.*
160. ^ *"SUSE Linux Enterprise Server 15 SP5: Chapter 17. UEFI (Unified Extensible Firmware Interface) (Administration Guide)"*. *documentation.suse.com. Retrieved 12 January 2025.*
161. ^ Jump up to: **a** **b** *"Linux on Windows 8 PCs: Some progress, but still a nuisance"*. *ZDNet. Retrieved 26 February 2013.*
162. ^ *"Lenovo UEFI Only Wants To Boot Windows, RHEL"*. *Phoronix. Retrieved 26 February 2013.*

163. ^ *"Linux acquitted in Samsung laptop UEFI deaths"*. *Bit-tech. Retrieved 26 February 2013.*
164. ^ *"Booting Linux using UEFI can brick Samsung laptops"*. *The H. Retrieved 26 February 2013.*

- *Zimmer, Vincent; Rothman, Michael; Hale, Robert (10 May 2007). "EFI Architecture". Dr. Dobb's Journal. UBM. Retrieved 12 October 2012.*
- *de Boyne Pollard, Jonathan (11 July 2011). "The EFI boot process". Frequently Given Answers. Retrieved 12 October 2012.*
- *de Boyne Pollard, Jonathan (8 December 2011). "The Windows NT 6 boot process". Frequently Given Answers. Retrieved 12 October 2012.*
- *Smith, Roderick W. (2011). "A BIOS to UEFI Transformation". Roderick W. Smith's Web Page. Retrieved 12 October 2012.*
- *Kothari, Rajiv (21 September 2011). "UEFI – Just How Important It Really Is". Hardware Secrets. Archived from the original on 25 October 2012. Retrieved 12 October 2012.*
- *Fisher, Doug (2011). "UEFI Today: Bootstrapping the Continuum". Intel Technology Journal. **15** (1). Intel. ISBN 9781934053430. Archived from the original on 27 September 2013. Retrieved 24 September 2013.*

- Official website
- UEFI Specifications
- Intel-sponsored open-source EFI Framework initiative
- Intel EFI/UEFI portal
- Microsoft UEFI Support and Requirements for Windows Operating Systems
- How Windows 8 Hybrid Shutdown / Fast Boot feature works
- Securing the Windows 10 Boot Process
- LoJax: First UEFI rootkit found in the wild, courtesy of the Sednit group
- Python Interpreter for UEFI Shell
