Unified Extensible Firmware Interface (UEFI)

Luciano Sánchez

Enero 2025

¿Qué es UEFI?

- **Definición:** UEFI (Unified Extensible Firmware Interface) es una plataforma de firmware para computadoras, diseñada para reemplazar el sistema BIOS tradicional.
- **Función Principal:** Su función es iniciar los componentes del sistema y cargar el sistema operativo desde el hardware al encender la computadora.
- **Beneficios Clave:** Proporciona arranques más rápidos, soporte para grandes capacidades de disco y una arquitectura más robusta para manejar el pre-arranque del sistema.

Reemplazo del BIOS Tradicional

- **Superando al BIOS:** UEFI ofrece una interfaz más moderna y flexible en comparación con el antiguo BIOS, que solo podía ejecutar código en modo real de 16 bits y tenía limitaciones de diseño inherentes.
- **Ventajas sobre BIOS:** Soporta discos de arranque mayores de 2 TB, proporciona inicializaciones más rápidas y es compatible con una arquitectura de sistema segura y extensible.
- **Compatibilidad:** Puede funcionar en modo de compatibilidad para soportar sistemas operativos antiguos que esperan un BIOS tradicional.

Orígenes de UEFI

- **Colaboración entre Intel y HP:** UEFI se originó en los años 90 como parte del desarrollo conjunto de Intel y Hewlett-Packard (HP) para los sistemas Itanium, que requerían capacidades de firmware no soportadas por el BIOS tradicional.
- **Problemas del BIOS:** Las limitaciones del BIOS incluían un modo de operación de 16 bits y un espacio de direcciones limitado a 1 MB, inadecuado para los sistemas avanzados en desarrollo en ese momento.

De EFI a UEFI

- **Introducción de EFI:** La primera versión de EFI (Extensible Firmware Interface) fue introducida por Intel para superar las limitaciones del BIOS y apoyar la arquitectura Itanium.
- **Evolución a UEFI en 2005:** EFI fue renombrada y ampliada a UEFI (Unified Extensible Firmware Interface) para enfatizar la estandarización y la adopción más amplia en la industria.
- **Establecimiento del UEFI Forum:** Intel junto con otros líderes tecnológicos fundaron el UEFI Forum para continuar el desarrollo y la promoción del estándar UEFI.

Expansión y Adopción de UEFI

- **Primera implementación de código abierto:** Tiano, lanzada por Intel en 2004, fue la primera implementación de código abierto de UEFI, permitiendo a la comunidad de desarrolladores participar en su evolución.
- **Adopción en la industria:** Desde su introducción, UEFI ha sido adoptado por numerosos fabricantes de hardware para mejorar la funcionalidad y seguridad del proceso de arranque en modernos sistemas de computación.
- **Innovaciones Continuas:** UEFI ha evolucionado para incluir características como Secure Boot y soporte para sistemas de arranque en discos de gran tamaño y múltiples formatos de tabla de particiones.

Especificación Abierta y Implementaciones Comunes

- **Especificación Estándar Abierta:** Mantenida por el UEFI Forum, una colaboración de empresas tecnológicas.
- **Implementaciones Comunes:**
  - **AMI Aptio:** Ampliamente adaptable, soporta una variedad de plataformas de hardware y es conocido por su flexibilidad en configuraciones de firmware. https://www.ami.com/aptio/
  - **Phoenix SecureCore:** Reconocido por incorporar avanzadas características de seguridad y por su confiabilidad en entornos empresariales.
    https://www.phoenix.com/phoenix-securecore-for-servers/
  - **TianoCore EDK II:** Proyecto de referencia de código abierto que sirve como una plataforma de desarrollo para futuras implementaciones de UEFI. https://www.tianocore.org

Servicios de Arranque y de Tiempo de Ejecución en UEFI

- **Servicios de Arranque:** UEFI proporciona un conjunto de servicios de arranque que se utilizan para inicializar el hardware, cargar los controladores, y cargar el sistema operativo. Estos servicios están disponibles hasta que el sistema operativo toma el control total del hardware y llama a la función ExitBootServices.
- **Servicios de Tiempo de Ejecución:** Después de que el sistema operativo se ha cargado, ciertos servicios proporcionados por UEFI siguen siendo accesibles, incluyendo servicios de tiempo, acceso a variables UEFI almacenadas de forma no volátil, y actualizaciones de la tabla de configuración del sistema.

Protocolo de Salida Gráfica (GOP)

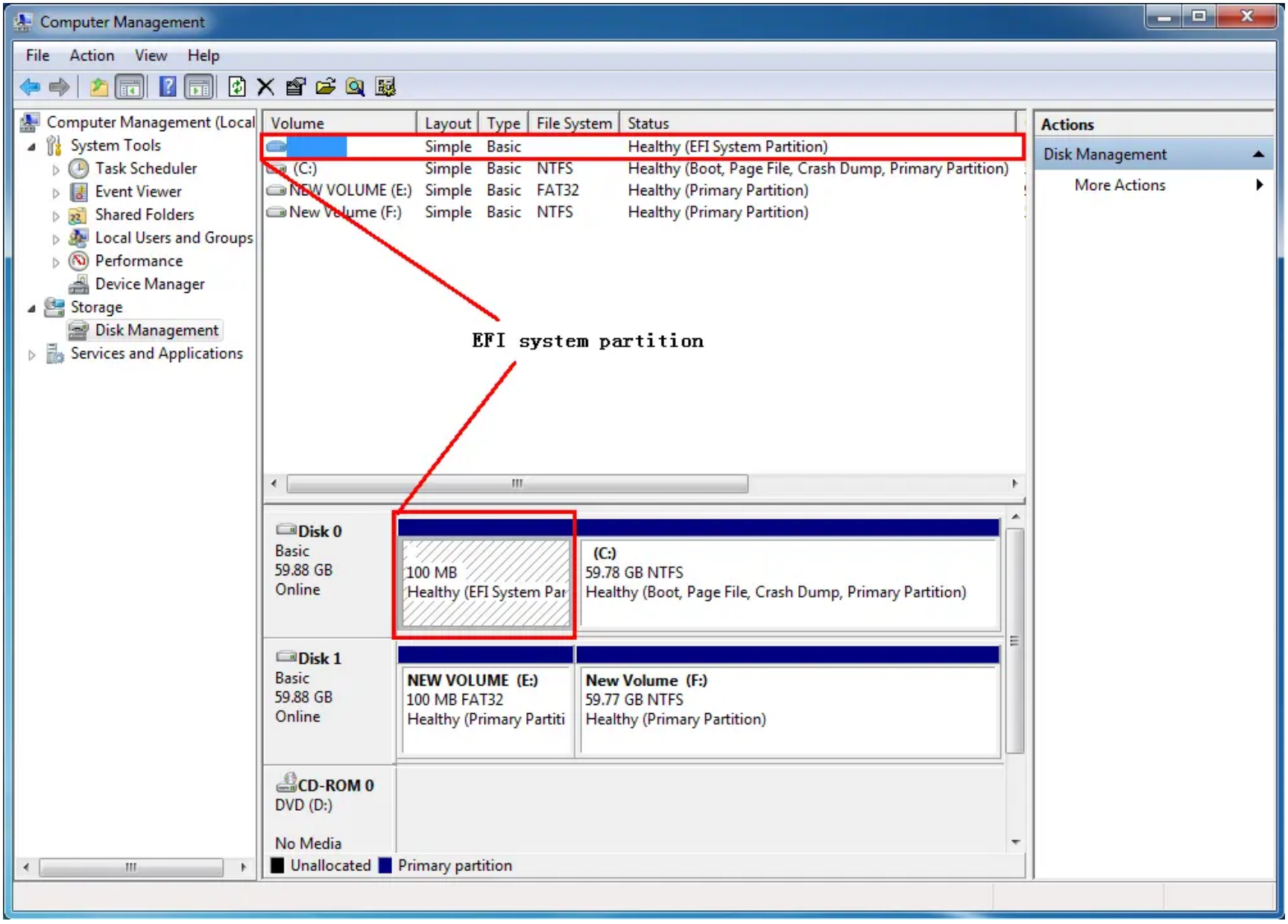
- **Función de GOP:** El Protocolo de Salida Gráfica define la interfaz entre el sistema operativo y el hardware de visualización del sistema, permitiendo que el sistema operativo controle las pantallas gráficas de alta resolución antes de que los controladores específicos del dispositivo estén cargados.
- **Capacidades de GOP:** GOP proporciona funciones básicas para la configuración de la pantalla, como la selección de modos de visualización y la renderización de gráficos primitivos, esencial para mostrar interfaces de usuario ricas en pre-arranque.

Mapa de Memoria UEFI y Servicios Variables

- **Mapa de Memoria UEFI:** UEFI mantiene un mapa de memoria que ayuda al sistema operativo a gestionar la memoria física del sistema.
- **Servicios Variables:** UEFI utiliza variables para almacenar información que puede ser utilizada tanto por el firmware como por el sistema operativo, como configuraciones de seguridad, opciones de arranque y datos del sistema.

Partición UEFI

- **¿Qué es una Partición UEFI?** La partición de sistema EFI (ESP) es una partición formateada con el sistema de archivos FAT32 en un disco utilizado en computadoras con UEFI. Es necesaria para almacenar los cargadores de arranque UEFI, las aplicaciones y los drivers necesarios para iniciar el hardware antes del lanzamiento del sistema operativo.
- **Propósito de la ESP:** Almacena los programas de UEFI que conectan el firmware UEFI con el sistema operativo y los dispositivos.
- **Requerimientos:** Debe ser una partición primaria y estar formateada en FAT32 para ser reconocida por UEFI.


Computer Management
File Action View Help
Computer Management (Local
System Tools
Task Scheduler
Event Viewer
Shared Folders
Local Users and Groups
Performance
Device Manager
Storage
Disk Management
Services and Applications
Volume Layout Type File System Status
Simple Basic Healthy (EFI System Partition)
(C:) Simple Basic NTFS Healthy (Boot, Page File, Crash Dump, Primary Partition)
NEW VOLUME (E:) Simple Basic FAT32 Healthy (Primary Partition)
New Volume (F:) Simple Basic NTFS Healthy (Primary Partition)
EFI system partition
Disk 0
Basic
59.88 GB
Online
100 MB
Healthy (EFI System Par
(C:)
59.78 GB NTFS
Healthy (Boot, Page File, Crash Dump, Primary Partition)
Disk 1
Basic
59.88 GB
Online
NEW VOLUME (E:)
100 MB FAT32
Healthy (Primary Partiti
New Volume (F:)
59.77 GB NTFS
Healthy (Primary Partition)
CD-ROM 0
DVD (D:)
No Media
Unallocated
Primary partition
Actions
Disk Management
More Actions

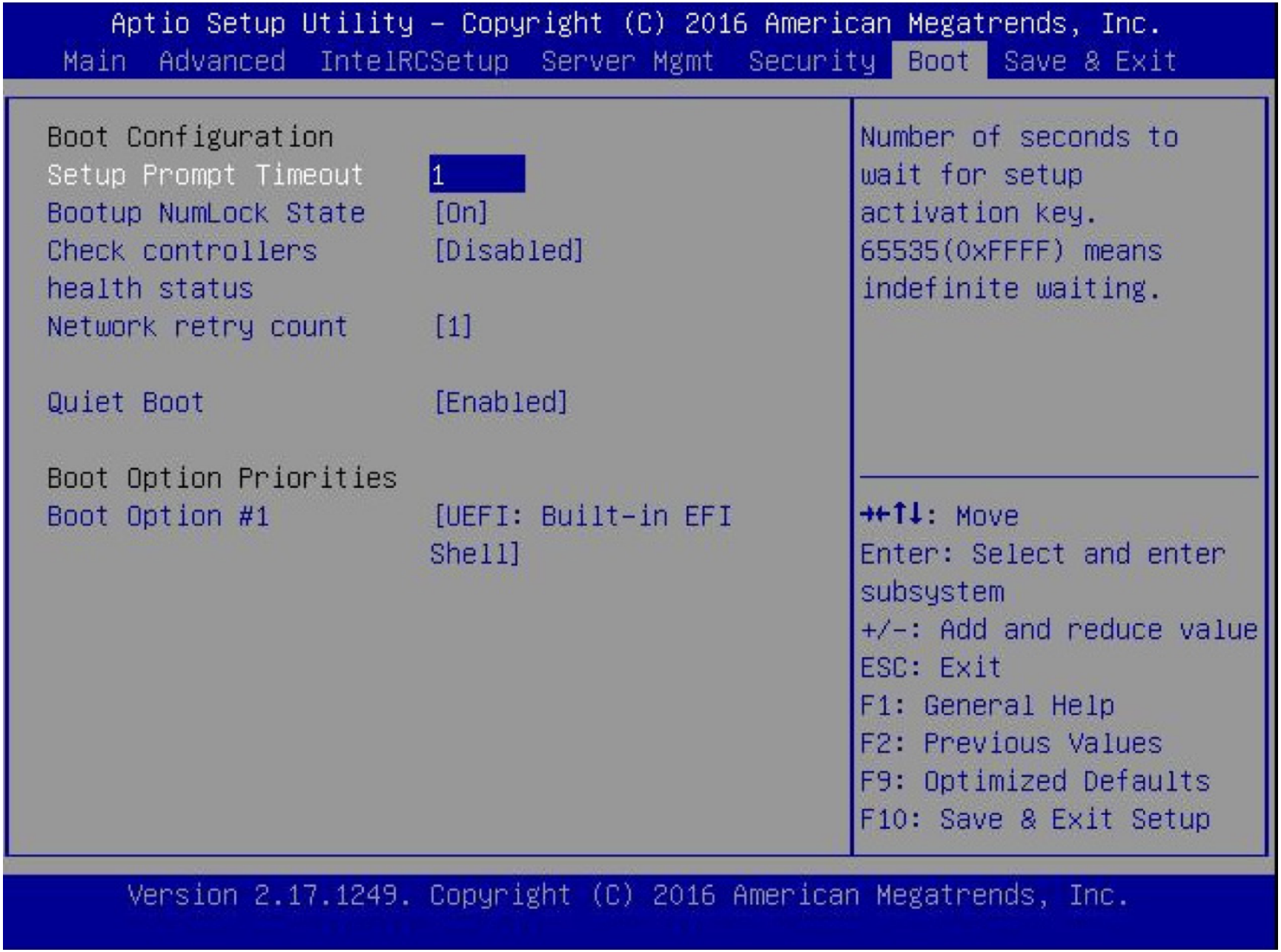
Estructura de Directorios en la Partición UEFI

- **Directorio Principal:** `/EFI/` - Contiene subdirectorios para cada sistema operativo o aplicaciones de firmware, como herramientas de diagnóstico o actualizaciones de firmware.
- **Cargadores de Arranque:** Por lo general, se encuentran en `/EFI/BOOT/` y `/EFI/Microsoft/Boot/` para sistemas Windows.
- **Archivos Importantes:**
  - `bootx64.efi` - El cargador de arranque por defecto para sistemas basados en x64.
  - `grubx64.efi` - Un ejemplo de GRUB como gestor de arranque para sistemas Linux.

```
/EFI
 |---- Microsoft
 |    |---- Boot
 |    |---- Recovery
 |---- Boot
 |    |---- bootx64.efi
 |---- Ubuntu
 |    |---- grubx64.efi
 |    |---- shimx64.efi
```

UEFI Shell

- Es un entorno de línea de comandos integrado en el firmware UEFI.
- Permite ejecutar comandos y scripts sin necesidad de un sistema operativo.
- Utilizado para administración, diagnostico y recuperación de sistemas.


Aptio Setup Utility - Copyright (C) 2016 American Megatrends, Inc.
Main Advanced IntelRCSetup Server Mgmt Security Boot Save & Exit
Boot Configuration
Setup Prompt Timeout
1
Bootup NumLock State
[On]
Check controllers health status
[Disabled]
Network retry count
[1]
Quiet Boot
[Enabled]
Boot Option Priorities
Boot Option #1
[UEFI: Built-in EFI Shell]
Number of seconds to wait for setup activation key. 65535(0xFFFF) means indefinite waiting.
→←↑↓: Move
Enter: Select and enter subsystem
+/-: Add and reduce value
ESC: Exit
F1: General Help
F2: Previous Values
F9: Optimized Defaults
F10: Save & Exit Setup
Version 2.17.1249. Copyright (C) 2016 American Megatrends, Inc.

Usos principales de UEFI Shell

- Gestion del arranque (`bcfg boot`)
- Recuperacion y reparacion de sistemas
- Exploracion y administracion de discos (`map -r`)
- Actualizacion de firmware y controladores
- Automatizacion con scripts NSH (`.nsh`)

Comandos basicos en UEFI Shell

- `help` - Muestra la lista de comandos disponibles.
- `map -r` - Lista discos y particiones disponibles.
- `fs0:` - Cambia a un disco especifico.
- `ls` o `dir` - Lista archivos y carpetas.
- `bcfg boot dump` - Muestra la configuracion de arranque.
- `bcfg boot add N fs0:\EFI\BOOT\BOOTX64.EFI "Mi SO"` - Agrega una nueva entrada de arranque.
- `reset` - Reinicia el sistema.

Como acceder al UEFI Shell

- Desde la configuracion UEFI (BIOS):
  - Entrar en la BIOS con `F2`, `F12`, `DEL` o `ESC` segun el fabricante.
  - Buscar la opcion "Launch EFI Shell".
- Desde un USB con UEFI Shell:
  - Descargar `Shell.efi` desde `https://github.com/tianocore/edk2`.
  - Formatear un USB en FAT32 y copiar `Shell.efi` a `\EFI\BOOT\BOOTX64.EFI`.
  - Arrancar el sistema desde el USB en modo UEFI.

Variables de UEFI y Gestión Remota

- **Variables de UEFI:** Son configuraciones almacenadas en la NVRAM que controlan el comportamiento del firmware, como las opciones de arranque y los parámetros de seguridad.
- **Modificación Remota:** Las variables pueden ser modificadas remotamente mediante herramientas de administración de sistemas, facilitando la gestión de múltiples servidores.
- **Ejemplos de Variables:**
  - `BootOrder` - Define el orden de arranque de los dispositivos.
  - `SecureBoot` - Habilita o deshabilita la funcionalidad de arranque seguro.

Configuración del Arranque de un Servidor desde Cero

- **Preparación del Disco:** Inicialización de un disco vacío utilizando una herramienta de partición para crear una Efi System Partition (ESP).
- **Formateo:** Formatear la ESP con el sistema de archivos FAT32.
- **Instalación del Gestor de Arranque:** Copiar el gestor de arranque EFI, como GRUB o Windows Boot Manager, en la partición ESP bajo el directorio adecuado (`/EFI/`).
- **Configuración del Firmware:** Acceder a la configuración UEFI para asegurar que el orden de arranque esté configurado para iniciar desde la partición ESP.

Secure Boot

- **Propósito de Secure Boot:** Secure Boot es una característica de seguridad en UEFI diseñada para asegurar que solo software confiable, verificado por firmas digitales, se cargue durante el proceso de arranque. Esto previene el arranque de aplicaciones no autorizadas y malware.

Secure Boot: Ejemplos Prácticos

- **Habilitando Secure Boot:**
  - Acceder a la configuración de UEFI en el arranque (usualmente presionando teclas como F2 o DEL).
  - Navegar hasta la sección de seguridad o arranque y seleccionar 'Secure Boot'.
  - Cambiar la configuración de 'Disabled' a 'Enabled'.
  - Ejemplo de comando en UEFI Shell: `setvar SecureBoot -guid ... -bs -rt -nv =%01%00`
- **Administración de Claves:**
  - Para añadir una nueva clave de plataforma (PK): `setvar PK -guid ... -bs -rt -nv -file myNewPK.cer`
  - Para verificar las claves instaladas: `dmpstore -guid ...`

Actualizaciones de Firmware UEFI con Ejemplos

- **Proceso de Actualización:**
  - Descargar la actualización de firmware desde el sitio oficial del fabricante.
  - Utilizar una herramienta verificada como ‘fwupdmgr‘ (en Linux) para aplicar la actualización.
  - Ejemplo de comando: `fwupdmgr update`
- **Rollback de Firmware:**
  - Asegurarse de que la opción de rollback esté habilitada en la configuración de UEFI.
  - Ejemplo de comando para revertir a una versión anterior: `fwupdmgr downgrade`

Protección contra Ataques de Firmware

- **Vulnerabilidades Comunes:** Los tipos comunes de ataques de firmware, incluyen ataques de sobreescritura, inyecciones de código y explotaciones de vulnerabilidades no parcheadas.
- **Estrategias de Mitigación:** El firmware UEFI se protege contra ataques mediante medidas de protección contra escritura, la utilización de módulos de plataforma confiables (TPM) y la implementación de políticas de seguridad.

¿Qué es TPM?

- **TPM (Trusted Platform Module)** es un chip de seguridad que protege claves criptográficas y verifica la integridad del sistema.
- **Funciones principales:**
  - Almacenamiento seguro de claves de cifrado.
  - Protección contra modificaciones en el firmware y arranque seguro.
  - Autenticación de hardware en redes y sistemas protegidos.
- **Versiones:**
  - **TPM 1.2:** Soporta solo algoritmos SHA-1, menos seguro.
  - **TPM 2.0:** Usa SHA-256 y algoritmos modernos, requerido para Windows 11.
- **Ejemplo de verificación de TPM en Linux:**

Comando

```
tpm2_getcap -c properties-fixed
```

Protección contra Ataques de Firmware con Escenarios

- **Escenario de Ataque de Sobreescritura:**
  - Implementar protección de escritura en el nivel de firmware para evitar modificaciones no autorizadas.
  - Utilizar TPM para validar la integridad del firmware en cada arranque.
  - Ejemplo de configuración segura de TPM 2.0:
    - Restablecer el TPM para asegurar un estado limpio: `tpm2_clear`
    - Establecer una contraseña de seguridad para la jerarquía del TPM: `tpm2_changeauth -c owner <nueva_contraseña>`
    - Verificar registros de arranque con PCR: `tpm2_pcrread`
- **Monitorización del Firmware:**
  - Usar herramientas como Chipsec para monitorizar la integridad del firmware y detectar modificaciones sospechosas.
  - Ejemplo de comando para verificar la integridad del BIOS: `chipsec_main -m common.bios_wp`
  - Verificación del estado de seguridad del TPM con: `tpm2_getcap -c properties-fixed`

¿Qué son los PCRs en TPM?

- **PCR (Platform Configuration Register)** es un registro en TPM que almacena hashes del proceso de arranque.
- Se usan para verificar la **integridad del firmware, bootloader y kernel**.
- **Funcionamiento:**
  - Cada fase del arranque genera un hash y lo extiende en un PCR.
  - Si un atacante modifica el arranque, los valores de PCR cambian.
- **Ejemplo de lectura de PCR en Linux:**

Comando

```
tpm2_pcrread
```

¿Qué es Chipsec?

- **Chipsec** es una herramienta de seguridad en firmware y hardware para analizar vulnerabilidades en el BIOS/UEFI.
- **Funciones principales:**
  - Detección de ataques al firmware y protección contra sobreescritura.
  - Análisis de configuraciones de seguridad en la plataforma.
- **Ejemplo de uso:**

Comando para verificar protección del BIOS

```
chipsec_main -m common.bios_wp
```

Integración con Herramientas de Gestión de Sistemas

- **Conectividad con Herramientas Externas:** Los scripts de UEFI Shell pueden conectarse con herramientas de gestión centralizadas como Microsoft System Center, **Ansible** o Puppet.
- **Automatización de la Configuración del Servidor:** También pueden usarse para desplegar configuraciones en múltiples máquinas a través de herramientas de gestión.
- **Ejemplos:**
  - Script para reportar la configuración del sistema a una herramienta de monitoreo central.
  - Automatización de updates de firmware a través de herramientas de gestión de IT.

Resolución de Problemas de Arranque en UEFI

- **Identificación de Problemas de Arranque:**
  - Pantallas negras sin error.
  - Mensajes como "No Bootable Device" o "Operating System Not Found".
  - Reinicios en bucle sin entrar al sistema operativo.
- **Métodos de Resolución:**
  - Verificar el orden de arranque en la configuración de UEFI.
  - Usar comandos en UEFI Shell para reparar la configuración de arranque.
  - **Ejemplo práctico: Cargar manualmente el gestor de arranque de Windows**
    - Listar particiones: `map -r`
    - Acceder a la partición UEFI: `fs0:`
    - Ejecutar el bootloader manualmente: `EFI\Microsoft\Boot\bootmgfw.efi`
    - Si funciona, restaurar la entrada de arranque con: `bcfg boot add 0 fs0:\EFI\Microsoft\Boot\bootmgfw.efi "Windows Boot Manager"`

Registro y Monitoreo con UEFI

- **Capacidades de Registro de UEFI:**
  - UEFI puede registrar eventos críticos del sistema, como fallos en el arranque, cambios en la configuración del firmware y errores de hardware.
  - Útil para la solución de problemas y auditorías de seguridad.
- **UEFI Event Log (Registro de Eventos):**
  - Almacena eventos como intentos de arranque fallidos, cambios en Secure Boot y modificaciones en variables NVRAM.
  - Puede accederse a través del firmware o herramientas como `fwupd` y `efibootmgr`.
- **Ejemplo de comando para ver entradas de arranque en Linux:**

Comando

```
efibootmgr -v
```

Despliegue de Sistemas a Gran Escala con UEFI

- **Automatización del Despliegue:** Uso de UEFI para automatizar la instalación y configuración de sistemas en data centers y entornos de nube. Implementación de scripts UEFI para estandarizar la configuración del firmware en múltiples máquinas.
- **Gestión Centralizada:** Integración de UEFI con plataformas de gestión de sistemas como Microsoft System Center o VMware vCenter para centralizar y simplificar el manejo de actualizaciones y configuraciones de firmware.
- **Ejemplo Práctico:** Configuración de un script UEFI para ejecutar un proceso de instalación de sistema operativo en red, aplicando configuraciones predefinidas a cada nuevo servidor desplegado.