Ozmosis BIOS Module

In diesem Post will ich jedem Ozmosis Nutzer das individualisieren des eigenen Ozmosis UEFIs ein wenig näher bringen und evtl. ein wenig die Augen öffnen, was das bearbeiten von Ozmosis ROMs angeht. Dabei werde ich darauf eingehen, wie ein jeder die Kern-Komponenten von Ozmosis identifizieren kann, den Inhalt von ROMs kürzen und verstehen kann und eigene Dateien hinzufügen kann... Vielleicht hilft das ja dem ein oder anderen Interessenten die Ozmosis Welt ein wenig besser zu verstehen 👍

Als erstes muss ein jeder Ozmosis Nutzer verstehen, dass auch das Ozmosis Paket für jeden PC angepasst werden muss, um perfekt zu funktionieren.
Dazu kann unter anderem zählen:

- Anpassen und individualisieren der **Defaults.plist**
- Installieren der **richtigen Kexts**
- Patchen von Kexts mit dem **KernextPatcher** (**KextToPatch** (**KernextPatcher**) sowie **ACPIPatcher**)
- Einrichten von **DarBoot** (ab APFS)
- Anpassen von ACPI Tabellen wie das **patchen der DSDT** und **erstellen von SSDTs**
- Auswählen des gewünschten **Themes**
- Hinzufügen von **UEFI Applikationen** wie der **HermitShell**

und so weiter...

Jede der oben aufgezählten Möglichkeiten kann dabei über die EFI gelöst werden (Wie das geht ist jeweils verlinkt). Sobald man sich das ganze jedoch ein bisschen genauer anschaut, erkennt man, dass all diese Dinge auch über das ROM passieren können.

Hierfür ein Beispiel warum dies eventuell sinnvoll ist:
Erstellt man eine individuelle Defaults.plist und setzt diese in die EFI in den Ordner Efi/Oz, wird diese nach einem NVRAM Reset genutzt. Aber woher stammen die SMBIOS Werte bevor man seine eigene Defaults.plist einsetzt?
Diese kommen aus einer Defaults.plist, die sich im BIOS befindet und in das installierte Ozmosis BIOS eingefügt wurde. Beim Einstellen einer Defaults in der EFI Partition, wird die im ROM befindliche Defaults.plist lediglich überschrieben/ersetzt und ist von da an im Ruhestand.

Bevor man jetzt alles doppelt moppelt, kann man nun also einfach, wenn man will, seine eigene Defaults ins ROM einsetzen und sie danach wieder aus der EFI entfernen.

Gleiches gilt auch für Kexts: Kopiert man eine aktuelle Version des FakeSMC nach Efi/Oz/Darwin/Extensions/Common, ersetzt diese lediglich die im ROM vorhandene FakeSMC oder SMCEmulator.kext.

Ebenfalls kann es für manche sinnvoll sein auf ein vollständiges BIOS zu setzen. Bei einem ROM mit zB.

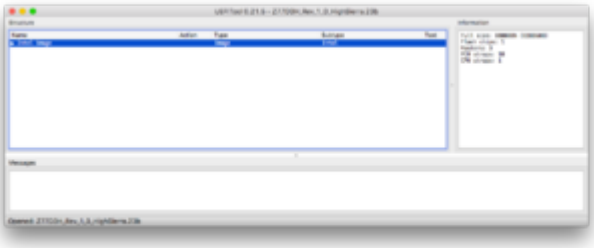
integrierter Ethernet Kext, kann schon während der Installation Ethernet für Anmeldungen, iCloud o.ä genutzt werden. Der Hackintosh ist komplett funktionstüchtig, selbst wenn auf einer neuen Platte installiert wird und dementsprechend keinerlei Dateien auf der EFI Partition liegen.

Das macht es vielleicht jetzt erstmal logisch sein ROM auf jeden Fall anzupassen, aber lasst mich euch warnen:
Das Kopieren aller existierenden Dateien aus dem EFI in das ROM, bringt keinen Vorteil was Performance oder ähnliches angeht! Im Gegenteil, das Einfügen möglichst vieler KernelExtensions in das BIOS, verlangsamt sogar den Boot Prozess!
Trotzdem bin ich ein Freund von Struktur und habe in meinem ROM gerne auch wirklich nur das was ich brauche. Dabei geht es vorallem um folgendes:

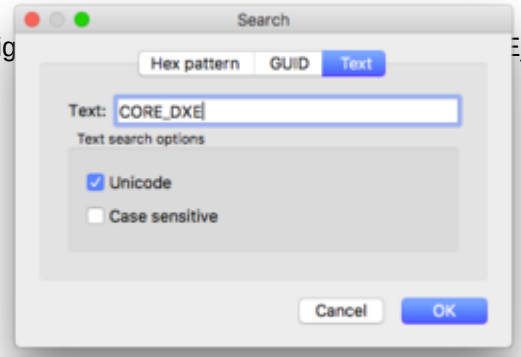
- Entfernen von ungenutzten Ozmosis Dateien und Kexts aus dem BIOS, um die Firmware nicht unnötig voll zu pressen
- Einfügen der für den eigenen Bedarf benötigten Tools. Dabei will der eine unbedingt eine Shell im BIOS, und der andere braucht einen KernextPatcher um Kexts und/oder Kernel zu patchen
- Ersetzen von Standard Konfigurationen mit der eigenen Konfiguration: Löschen der standardisierten Defaults.plist, einfügen der eigenen Defaults.plist

Aber fangen wir doch einfach mal an!
Ich werde euch in diesem ersten Guide nicht zeigen, wie ihr euer eigenes BIOS von Vanilla zu einem Ozmosis BIOS macht, das ist ein anderes Thema (Wer genau aufpasst sollte aber nach den Guides fähig sein, sein eigenes ROM aus einem original BIOS zu schaffen). Deswegen nehmen wir als Basis einfach die vorgefertigten ROMs aus der Downloadsektion... Jedes BIOS sieht anders aus, aber alle haben bestimmte Komponenten für Ozmosis enthalten, je nach Platz, mehr oder weniger. In diesem Fall werde ich ein Z77 BIOS verwenden, da in diesen ROMs viel Platz und dementsprechend auch die größte Anzahl an Ozm-Komponenten zu finden ist. Das ganze lässt sich jedoch danach auf andere ROMs übertragen...

Für unsere Aufgabe eignet sich am aller besten das UEFI-Tool, da dieses das ROM in einer sehr übersichtlichen GUI darstellt und uns automatisch vor Fehlern bewahrt.
UEFI-Tool installiert und gestartet müssen wir einfach nur unser Ozm BIOS aus der Datenbank nehmen und auf das UEFI Fenster per Drag und Drop ziehen. Alternativ geht dies auch über Menüleiste-->File-->Open Image File (All files).
Ist dies passiert wird uns der Inhalt des ROMs präsentiert. Das kann dann zB so aussehen:

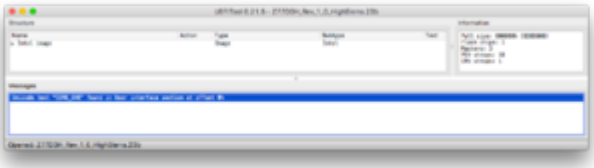


Das ganze ist eine Ordnerstruktur wie man sie vom Finder kennt. Es gibt mehrere Firmware Volumen für unterschiedliche Zwecke mit unterschiedlichen Größen, aber das soll uns alles garnicht interessieren, denn uns interessiert nur alles was Ozmosis betrifft! Beim anfertigen eines Ozmosis ROMs, wird lediglich am richtigen Ort im ROM ein wenig Platz geschaffen und daraufhin verschiedenste Dateien, zuständig für den erfolgreichen Boot von OS X eingefügt. Diese Datein kommen immer an die gleiche Stelle und auch nur diese Stelle interessiert uns. Dabei handelt es sich um das Volumen in dem die Datei namens `CORE_DXE` zu finden ist. Bevor wir jetzt aber das durchkämmen anfangen, suchen wir einfach danach.
Mit CMD/Win+F öffnet sich ein Suchfenster. Hier muss jetzt die Spalte "Text" ausgewählt werden, die

richtig 

E_DXE gesucht werden. Das sieht dann so aus:

Bestätigen wir das ganze mit OK, erhalten wir einen Eintrag in "Messages":

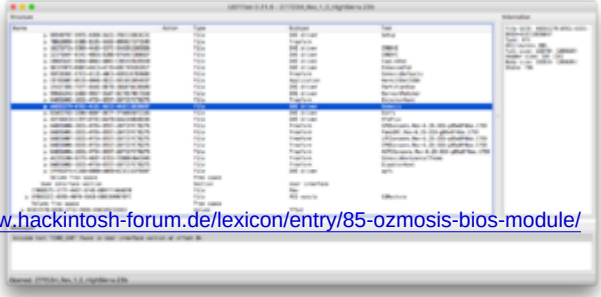


Ein einfacher Doppelklick auf den Eintrag bringt uns auch direkt an die gewünschte Stelle, und das sieht so aus:

Jeder einzelne Eintrag ist ein Firmware-Modul, das sich im BIOS befindet. Die Einträge tragen:

- eine GUID: die lange Nummer, jede Nummer kommt nur einmal im ganzen ROM vor
- eine Art des Verzeichnis (Type): Region, Volume, File, Section, FreeSpace etc.
- eine bestimmte Dateiart (Subtype): Wie zB Treiber (DXEDriver), Programme wie die Shell (Application), undefinierte Freiformen (Freeform) etc.
- und meistens einen DateiName (Text): Dieser text sagt meistens um was es sich genau handelt und ist das am einfachsten identifizierbare Merkmal

Soeben haben wir nach dem Text `CORE_DXE` gesucht und landen dementsprechend bei einer Datei mit dem Namen CORE_DXE. Im gleichen Volumen wie diese Datei, befinden sich alle für Ozmosis relevanten Dateien. Häufig sind diese dabei am Ende des Abschnitts zu finden. Wenn wir also runterscrollen werden wir meistens direkt mit einigen Dateien konfrontiert, die je nach ROM unterschiedlich sind, **eine von ihnen mit dem Namen `Ozmosis`**
So sieht das ganze aus in meinem Beispiel:



In meinem Beispiel gehören hier ziemlich viele Dateien zu Ozmosis (von `Enhanced Fat` bis `apfs`), die ich im folgenden aufzählen und kurz beschreiben werde:

- `Ozmosis`: Dies ist der KernTreiber in dem das Hexenwerk passiert und der für das meiste zuständig ist. Dieser Treiber braucht manchmal Updates, genauso wie Clover updates braucht, um mit neuen macOS Versionen zu funktionieren. Diese Updates sind aber **viel** seltener als bei Clover! **In jedem BIOS enthalten!**
- `OzmosisDefaults`: Ganz einfach: Eine Defaults.plist, konvertiert in eine Version die ins BIOS kann! Bei den meisten wird es sich hier um eine standardisierte Version handeln, die in den ROMs aus unserer DownloadSektion untergebracht wird. **In jedem BIOS enthalten!**
- `EnhancedFat`: Ein Treiber, der das Benutzen von FAT Dateisystemen ermöglicht. So kann an erster Stelle unsere EFI gelesen und von Ozmosis verarbeitet werden. **In jedem BIOS enthalten!**
- `HFSPlus`: Ebenfalls ein FileSystem-Treiber, der das Verwenden von HFS Platten ermöglicht. HFS wird von Apple schon lange genutzt und wurde nun von APFS und HighSierra ersetzt. HighSierra kann jedoch nach wie vor mit HFS benutzt werden, weshalb dieser Treiber auch für HighSierra sehr wichtig ist. **In jedem BIOS vor Mojave enthalten!**
- `apfs`: Der Apple FileSystem DateiSystem-Treiber. Dieser ermöglicht das Benutzen von APFS Platten und muss in jedem High Sierra ROM enthalten sein.
  Alternativ kann der ApfsDriverLoader genutzt werden (am besten mit Kext2FFS in ffs umwandeln) dieser ist wesentlich kleiner und lädt die zum OS passende APFS Version. **In jedem BIOS ab High Sierra enthalten!**
- `FakeSMC`, `SMCEmulator` oder `VirtualSMC`: Dies ist eine weit bekannte KernelExtension die den Apple SMC simuliert. Entweder FakeSMC **oder** SMCEmulator **oder** VirtualSMC muss in jedem BIOS enthalten sein. **In jedem BIOS enthalten!**

All diese Dateien sind also in jedem ROM zu finden! Sie sind die Minimalkonfiguration eines mit Ozm ausgestatteten BIOS!

Alle folgenden Dateien sind "freiwillig" und werden je nach Platz im ROM mit untergebracht.

- `OzmosisHorizontalTheme`: Das Theme, auch bekannt als GUI oder UserInterface, das man sich beim Start des PCs als Bootauswahl anzeigen lassen kann. **Für Multiboot-Systeme stark empfohlen.**
- `DarBoot`: Ein kleines Programm von Cecekpawon, dass dafür sorgt, dass alle macOS Booteinträge angezeigt werden und auch APFS Booteinträge bei einem NVRam-Reset nicht verloren gehen. **Empfohlen für alle Systeme ab und einschließlich High Sierra.**
- `KernextPatcher`: Cecekpawon's Kext und Kernel Patcher, der Kernel- und KextsToPatch Einträge über eine KernextPatcher.plist in /Efi annehmen kann.
  Der KernextPatcher ist in einer speziellen Form in jedem Mojave-kompatiblen ROM enthalten!
- `AcpiPatcher`: Cecekpawon's ACPI und DSDT Patcher, der ACPI-Renames über eine AcpiPatcher.plist in /EFI annehmen kann und ebenfalls CPU-SpeedStep-States generieren kann.
- `ExtFS`: Das Ext Dateisystem, das viel bei Linux genutzt wird. Wer kein Linux nutzt, braucht dies im Normalfall auch nicht.
- `HermitShellX64`: Eine Variante des UEFI Shell Programms, über das Änderungen am System gemacht werden können. Das Programm kann unter anderem auf den NVRAM und die Dateien eines Systems zugreifen, wodurch zB. Veränderungen an der EFI vorgenommen werden können. Häufig erscheint diese als Eintrag "Built-In Shell" in der Bootauswahl des Mainboards.
- `GPU-, LPC-, ACPI-, CPUSensors`: Die FakeSMCSensoren, die häufig gerne im BIOS untergebracht werden. Mit ihnen lässt sich unter anderem die Hardware überwachen/überprüfen. Dabei können zB mit dem OS X Programm "HWMonitor", Temperaturen, Taktraten und Fan-Geschwindigkeiten ausgelesen werden.
- `PartitionDXE`: Treiber, der für die Unterstützung von ungewöhnlichen Partitionstabellen wie Apple Partition Map oder hybrid GPT/MBR sorgt. Dieser Treiber ist überwiegend für SnowLeopard Installationen und nicht für neuere macOS notwendig!

- `InjectorKext`: Kernel Extension, notwendig für den Boot von SnowLeopard. NEIN! Diese Kext ist heutzutage nicht für das Injecten von Kexts zuständig! Man braucht sie lediglich für SnowLeo.
- `DisablerKext`: Kext die das Power Management für Mac OSX SnowLeopard deaktiviert. Auch dieser Kext ist für neuere OS X nicht notwendig!
- `VoodooHDA`: Veraltete Kext für AudioLösungen. Diese ist in den meisten ROMs nicht mehr zu finden, da mittlerweile überwiegend auf andere Lösungen gesetzt wird.
- `Andere Kexts`: *Prinzipiell lässt sich jede Kext ins ROM integrieren. Dementsprechend ist es auch möglich das man hier zB EthernetKexts wie Realtek oder IntelMausi findet, jedoch auch USBInjectAll und alles sonst noch denkbare.*

Ihr seht, es gibt sehr viele Möglichkeiten zu einem BIOS, von der minimal Konfiguration abweichende Dateien hinzuzufügen... Umso schwerer ist es also für die Ersteller der ROMs einen Mittelweg zu finden, um jeden User happy zu machen. Das ganze geht aber nicht immer so, und während der eine das Theme vermisst, hat jemand anderes ein ungenutztes Theme und vermisst die Shell.
Deswegen will ich euch zeigen wie ihr euer BIOS anpassen könnt um euren Ansprüchen gerecht zu werden und nebenbei die Funktionsweise der einzelnen Komponenten erklären, damit dem ein oder anderen ein paar Hintergründe klar werden. Vielleicht weckt die Sache ja auch so euer Interesse, dass ihr gerne selber für andere Ozmosis fähige ROMs bauen wollen würdet, denn das ganze ist eigentlich wirklich kein Hexenwerk.

Hier erfährst du, wie du dein Ozmosis ROM individuell anpassen kannst: Ozmosis BIOS personalisieren