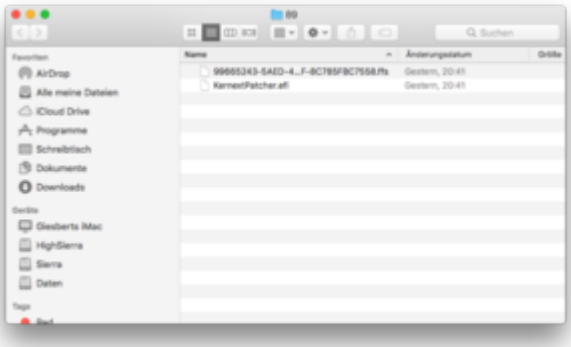
Erledigt On The Fly Kernel und Kextpatcher für Ozmosis

Beitrag von „griven“ vom 14. Juli 2017, 00:06

Einer der größten Kritikpunkte an Ozmosis ist die Tatsache das es mit Bordmitteln von Ozmosis bisher nicht möglich ist Kernel und Kextpatches ähnlich wie bei Clover on the Fly zu realisieren sprich wo immer Patches nötig sind (Whitelist Patch für Broadcom WLAN Karten, NVME SSD usw...) war man mit Ozmosis bisher außen vor denn die Patches ließen sich nicht auf eine ähnlich komfortable Weise wie mit Clover ins System einschleusen sondern man musste entweder selbst Hand anlegen und einen HEX Editor bemühen oder aber eine Litanei von Perl Befehlen ausführen beides keine wirklich zufrieden stellende Lösung da hierbei die Dateien dauerhaft verändert werden und man das Prozedere nach jedem Update wiederholen durfte bis jetzt...

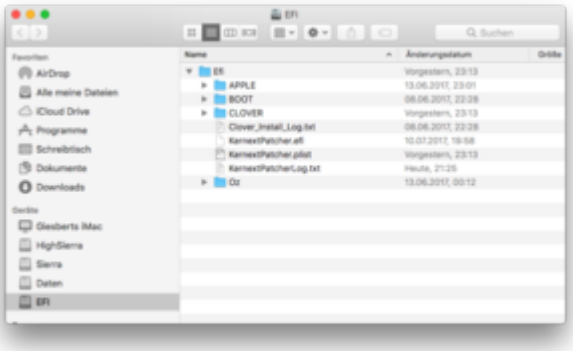
@cecekpawon hat sich diesem Problem angenommen und Abhilfe in Form eines Kernel and Kextpatchers geschaffen der sich sowohl als Driver über die UEFI Shell (nicht ganz unwichtig wenn wenig Platz im ROM ist) als auch als FFS direkt im ROM unterbringen lässt. Das Ganze greift das ExitBootServices Event ab und klemmt sich lax gesprochen zwischen OZ und den Start des Kernel durch OS-X und nimmt ähnlich wie Clover die Patches dann im RAM vor (innerhalb des prelinked Kernels) der Vorteil liegt auf der Hand da diese Methode nicht destruktiv ist sprich es werden keine Files auf der Platte verändert aber wie funktioniert das nun und was muss getan werden um es zu nutzen?

Als erstes braucht man logischerweise erstmal den Patcher selbst den man sich am besten auf der dazu gehörenden GitHub Seite besorgt (https://github.com/cecekpawon/...ter/Module/KernextPatcher). Einmal geladen befindet sich im Downloads Ordner ein Verzeichnis mit folgendem Inhalt



Je nachdem wie man den Patcher einsetzen möchte benötigt man aus dem Verzeichnis entweder die .efi (als Treiber über die EFI Shell geladen) oder die .ffs (direkt in den ROM

gebaut) Datei. Ich empfehle aktuell den Treiber noch über die EFI Shell zu laden denn er befindet sich aktuell noch in der Entwicklung und durch das laden über die EFI Shell lassen sich die Versionen ggf. schnell austauschen ohne dabei jedes mal den ROM neu flashen zu müssen. Damit der Treiber über die EFI Shell geladen werden kann muss er auf die EFI Partition im Ordner EFI abgelegt werden



zusätzlich zum Patcher selbst braucht es natürlich auch noch das File in dem die einzelnen Patches definiert werden müssen. Dieses File (KernextPatcher.plist) ist eine einfache .plist die sich an der Struktur der Kext2Patch Einträge aus der config.plist von Clover orientiert (besonders praktisch für Clover Umsteiger) hier mal ein Beispiel wie die Datei aussehen könnte:

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
<key>KernextPatches</key>
<dict>
<key>KernelToPatch</key>
<array>
<dict>
<key>Comment</key>
<string>Reboot fix for xcpm</string>
<key>Disabled</key>
<false/>
<key>Find</key>
<data>
VUiJ5UFXQVZBVUFUU1BBidZBifdliftFhf8PhA==
</data>
<key>MatchOS</key>
<string>10.12</string>
```

```
<key>Replace</key>
<data>
w0iJ5UFXQVZBVUFUU1BBidZBifdIiftFhf8PhA==
</data>
</dict>
<dict>
<key>Comment</key>
<string>xcpm bootstrap</string>
<key>Disabled</key>
<false/>
<key>Find</key>
<data>
g8PEg/si
</data>
<key>MatchOS</key>
<string>10.12</string>
<key>Replace</key>
<data>
g8PGg/si
</data>
</dict>
<dict>
<key>Comment</key>
<string>xcpm bootstrap</string>
<key>Disabled</key>
<false/>
<key>Find</key>
<data>
idgExDwidyI=
</data>
<key>MatchOS</key>
<string>10.13</string>
<key>Replace</key>
<data>
idgExjwidyI=
</data>
</dict>
<dict>
<key>Comment</key>
<string>Reboot fix for xcpm</string>
<key>Disabled</key>
<false/>
```

```
62. <key>Find</key>
63. <data>
64. vgsAAABd6QgAAAAPH4QAAAAAAFVIieVBVw==
65. </data>
66. <key>MatchOS</key>
67. <string>10.13</string>
68. <key>Replace</key>
69. <data>
70. vgsAAABd6QgAAAAPH4QAAAAAAMNIieVBVw==
71. </data>
72. </dict>
73. </array>
74. <key>KextsToPatch</key>
75. <array/>
76. </dict>
77. <key>Preferences</key>
78. <dict>
79. <key>Debug</key>
80. <false/>
81. <key>Off</key>
82. <false/>
83. <key>SaveLogToFile</key>
84. <true/>
85. </dict>
86. </dict>
87. </plist>
```

Alles anzeigen

Ihr seht der Aufbau entspricht der Kext2Patch bzw. Kernel2Patch Sektion der config.plist von Clover es lassen sich sogar die Einträge aus der config.plist von Clover 1:1 in dieses Konstrukt übernehmen. Um das ganze nu zu nutzen muss der Patcher dem UEFI nun noch vertraut gemacht werden sofern Ihr nicht das .ffs direkt in den ROM gebaut habt sondern wie empfohlen den Weg über die EFI Shell geht kommt ihr mit den folgenden Befehlen ans Ziel.

1. EFI Shell starten -> Bios Bootmenu -> Buildin EFI Shell.
2. Filesystem mittels fs0: auf die EFI Partition der ersten Platte wechseln.
3. Mittels cd EFI ins EFI Verzeichnis wechseln.
4. Mit dem Befehl bcfg driver add 0 KernextPatcher.efi "KernextPatcher" den Treiber hinzufügen.
5. Exit eingeben und neu starten

Wenn Ihr alles richtig gemacht habt verrichtet der Patcher ab jetzt seinen Dienst und protokolliert sein Tun in einem Logfile auf der EFI Partition. Sobald Ihr alle Patches die Ihr benötigt in die KernextPatcher.plist eingefügt habt und alles zu Eurer Zufriedenheit läuft könnt Ihr im unteren Bereich der KernextPatcher.plist das Logging auch abschalten hierzu einfach im Bereich Preferences der KernextPatcher.plist den Wert für den Key SaveLogToFile auf false setzen.

So nun aber viel Spaß beim testen und natürlich heißen Dank an @cecekpawon für dieses geniale Ding