Funktionierender NVRAM nun mit Clover

Beitrag von „mhaeuser“ vom 14. Januar 2018, 22:06

Ja, NVRAM nutzt AppleEFIRuntime. Welche funktion ruft die App auf?

EDIT: Nach 'nem normalen Boot oder nach S4-Wake?

EDIT2: Ich gehe einfach mal davon aus, dass es eine Variable-Funktion ist, die wir nicht überschreiben... also, das Problem ist, dass ein Teil des AMI-Flash-Treibers vom NVRAM-SMM-Treiber beschrieben wird. Mit dem alten AptioFix wurde diese Speicheradresse physisch verschoben und natürlich auch woanders gemappt, weswegen der NVRAM-Treiber dann fröhlich ins Nirvana schreibt, was zufälligerweise in einem R/W-Speicherteil erfolgt. Mit dem neuen AptioFix wird das physische Verschieben verhindert und der SMM-Treiber schreibt in einen eigentlich R/O-Bereich, was zu einem GPF-KP führt. Deshalb wurden die drei Variable-Funktionen, GetVariable, SetVariable und GetNextVariableName überschrieben, um beim Aufruf das WP-Bit aus dem CR0 rauszuhauen, was den KP verhindert. Wenn du eine andere, nicht überschriebene Funktion aufrufst, führt das erwartungsgemäß zu einem KP.
Die Entscheidung, nur diese drei zu überschreiben, war strategisch, da macOS nur diese drei Var-Funktionen nutzt. Das WP-Bit ist leider Core- und nicht Thread-bezogen, deshalb kann ohne einen Spinlock, wie AppleEFIRuntime ihn setzt, der zweite Thread im Kern auf XNU-Ebene in R/O-Speicher schreiben. Wenn eine Kext also einen Bug oder was auch immer hat, dass sie out-of-bound schreibt und zufällig der andere Thread gerade einen RT-Dienst mit dem WP-Code ohne Spinlock (also an AppleEFIRuntime vorbei) aufruft, wird das WP-Bit rausgekickt und die Kext schreibt erfolgreich ins Nirvana.
Deswegen: Seit wann gibt es dieses User-Client-Interface für AppleEFIRuntime? Mir wurde gesagt, dass es sowas nicht gibt... mit der Nutzung von diesem gibt es keinen Grund, einen RT-Dienst direkt aufzurufen und das obige Problem ist gelöst.