OpenCore-Version ohne Booten bestimmen

Beitrag von „HAI“ vom 20. August 2021, 02:28

Ist ziemlich lästig, wenn man erst booten muß, bei einer anderen EFI, um die Version zu bestimmen und auch ziemlich fehleranfällig.

Bei einer "übergebenen" EFI hier im Forum kann dann sehr schnell die Version ermittelt werden.

Hier ist eine 100% Methode.

Von jeder Version **hier genau einmal** den Hash zentral ermitteln (sha-256). Von der Datei "opencore.efi" im EFI-Verzeichnis.

Hier schon einmal die 7er plus die historische 0.0.1.

Code

1. CA70EE4DE2B7365795A9FF25E9686D134B6C2952457F02EC2A98D85D3A4A2313 - 0.0.1

Code

1. 5984E2A40124CCCCE2871D755188BBD21A0CEE0A3D15A2EE9E46936D5086D36F - 0.7.2

Code

1. EBB13921259DE09C71927B6D8705B521521A5A3C2FFF0E1C28E552C2AFD7BF39 - 0.7.1

Code

1. CA33FD96EA1C088A54E17123F4A3D42B8D68E4DFFA4A138B070B93AE8774AAC1 - 0.7.0

Diese Methode funktioniert auch, wenn sich dort das Team entschließt, bei einer Version keine Änderung der zentralen "opencore.efi" vorzunehmen. Genau eines wird sich dann ändern, das

ist die Versionsnummer. Diese kleine Änderung bewirkt eine starke Veränderung des hashes, das ist eine Eigenschaft einer solchen Prüfsumme. Vorne und hinten 3 Stellen zur Prüfung reichen aus. Man muß nicht alle 64 vergleichen.

Damit bestimmt man die Versionsnummer, wie im Hackintool beim Booten. An einer funktionierenden EFI bleibt selbstverständlich noch eine Menge zu tun.

Hier noch die zentrale Datei für die Versionsnummer zur Veranschaulichung.

```
#include <Protocol/OcBootstrap.h>

/**
  OpenCore version reported to log and NVRAM.
  OPEN_CORE_VERSION must follow X.Y.Z format, where X.Y.Z are single digits.
**/
#define OPEN_CORE_VERSION           "0.7.3"

/**
  OpenCore build type reported to log and NVRAM.
**/
#if defined (OC_TARGET_RELEASE)
#define OPEN_CORE_TARGET            "REL" ///< Release.
#elif defined (OC_TARGET_DEBUG)
#define OPEN_CORE_TARGET            "DBG" ///< Debug with compiler optimisations.
#elif defined (OC_TARGET_NOOPT)
#define OPEN_CORE_TARGET            "NPT" ///< Debug with no compiler optimisations.
#else
#error "Unknown target definition"
#endif

```

Sascha_77 möchte noch bitte Deinen Stempel drauf geben. Kann das Hackintool zwischen "DBG" und "REL" unterscheiden (Log wahrscheinlich schon)? Debug hat eine andere Prüfsumme, dann hätten wir genau zwei für ein Release.

Für die Bestimmung dieser Prüfsumme benötigt man etwa 17 Sekunden.

Das ist das Drag and Drop Tool (GUI). Für die commandliner "shasum".

https://www.quickhash-gui.org/

(Bei wichtiger Software, wie z.B.: Linux, wird das ebenfalls gemacht (xxxx.iso). Zusätzlich gibt es dann eine digitale Unterschrift. Die benötigen wir nicht.)

(Man kann natürlich alle Versionen downloaden, d.h. dann auch die Debugs. Dann Zeitstempel und Größe vergleichen. Die Prüfsummenlösung ist einfacher und schneller.)

Beitrag von „Raptortosh“ vom 22. August 2021, 11:32

Das kann man so machen, aber es gibt von jeder oc Version viele Nightly Versionen. Für jede dieser Versionen wäre ein anderer hash Wert nötig, um auch diese erkennen zu können. So kann man nur die finale Version damit bestimmen, aber, wenn eine Nightly genutzt wird, funktioniert das nicht, außer man hätte wirklich den hash Wert jeder oc Nightly...

Beitrag von „Sascha_77“ vom 22. August 2021, 22:23

@__.__

Ist nicht das Hackintool, ist der Kext Updater der mir gehört.

Und der kann nur unterscheiden wenn das "Sicherheitsfeature" die Versionsausgabe ins NVRam zu unterdrücken abgeschaltet ist. Und sich an den Hashes orientieren ist mir zu aufwändig. Da müsste ich ja permanent eine Art Datenbank mit den ganzen Hashes vorhalten und pflegen. Steht für mich in keinem guten Zeit/Nutzen verhältnis. Und wenn die Leute sich dann stellenweise ihr OC noch selber klöppeln ists mit der Hashdatenbank sowieso essig.

Zwischen Release und Debug kann der KU ebenfalls unterscheiden. Aber nur wenn, wie oben schon geschrieben, dass Sicherheitsfeature deaktivert ist.

Beitrag von „HAI“ vom 23. August 2021, 06:55

Raptortosh

Sascha_77

Das mit den "nightlys" habe ich bei mir wieder rausgestrichen, weben tl,dr. Folgendes, wir haben die Summen für die festen (REL/DEB).

Wer nightlys verwendet, **weiß,** daß er den Release-Pfad verläßt. Diese Summen werden nicht verwaltet. Der Anteil derer ist klein.

Damit erkennt man an der Prüfsumme sofort, daß er kein "Release" verwendet. Das teilt er dann auch mit. Jeden Tag eine Neue ...

Es geht hier um die große Masse und nicht um ein paar Ausnahmen.

Jeder kann auch für sich selbst kontrollieren. Oft liegen ein paar Kopien rum und man weiß nicht welche Version die haben.

Man hat diese Summen für die Releases. Das sind nicht viele. Zu alte würde ich auch rauslassen, die werden dann ebenfalls nicht erkannt. Damit kann man sowieso nichts mehr sinnvolles anfangen.

Man muß die auch nur einmal hier zentral pflegen. Nicht jeder, Sascha ist doch viel einfacher.

Das ist eine einfache und schnelle Versionserkennung auch für die Supporter. Die Zusammenstellung der EFI ist dann die Hauptarbeit aber nicht das Thema hier.

(Das ist die Prüfsumme genau einer Datei, opencore.efi. Der Vorschlag war nicht eine Prüfsumme über das komplette EFI-Verzeichnis.)

Beitrag von „ozw00d“ vom 23. August 2021, 07:42

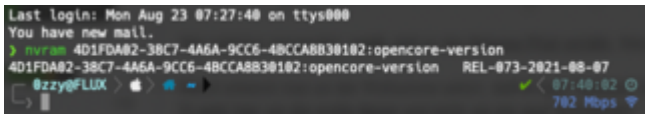
HAI ohne booten reicht es die aktuelle version via GitHub zu bestimmen. Dafür benötigst du weder einen Hashwert noch irgendwas spezielles.

Nach dem Boot ist das etwas anderes, bin ich auch so ein Nightly User, überprüfe ich es immer mit folgendem Command im Terminal:

Code

1. nvram 4D1FDA02-38C7-4A6A-9CC6-4BCCA8B30102:opencore-version

Ausgabe dann eben so:



Beitrag von „HAI“ vom 23. August 2021, 08:36

Habe nachgefragt, ob das über die "nvram"-Lösung der gebooteten Version machbar ist. Nun hast Du ja die Lösung. Im Hackintool bei "NVRAM" sieht man die Version nicht? Ok, Du siehst,

daß es eine nightly (Version plus 1) ist, auch Details?

Kannst Du bitte noch die "Github" Möglichkeit erläutern? Bis jetzt sehe ich da nur ein "Ausprobieren" indem man die Versionen durchgeht.

Wenn alles einfach geht, wird man definitiv Eintrag eins anpassen. Das mache ich selbstverständlich.

Beitrag von „ozw00d“ vom 23. August 2021, 16:26

HAI kein ding.

Die aktuellen rdy2use releases (so nenn ich das mal) findest du immer >>hier<< auf GitHub.

auf der GitHub Seite etwas mittig findest du dann die Anzahl der aktuellen Releases (siehe screenshot).

was dann insgesamt auf aktuell 27 releases hinausläuft.

Nightlys sind quasi der snapshot des aktuellen Standes vom aktuellen tag.

Ich hoffe das klärt es ein wenig auf, weshalb man sich ein "herausfinden" aufgrund der korrekt heruntergeladenen version sparen kann.

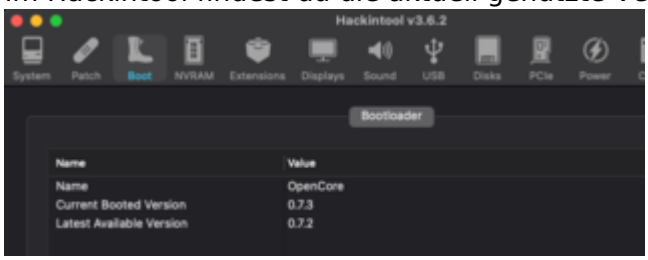
Wenn du ein wenig mit git arbeitest, es reicht wenn du verstehst was ein "git clone" bewirkt,

wirst du sehen das auch eine Versionierung auf dem eigenen Lokalen Storage ohne weiteres möglich ist.

Geht mal etwas schief, lädst du dir ganz einfach das release womit du zuletzt gearbeitet hast herunter und alles ist wieder hübsch.

Nachtrag:

Im Hackintool findest du die aktuell genutzte Version unter Boot:



Beitrag von „HAI“ vom 23. August 2021, 20:32

ozw00d

Dein Beitrag wie immer ist sehr hilfreich. Versuche nochmal zusammenzufassen und etwas hinzuzufügen.

Die Version für das nightly bei Dir oben hat offensichtlich noch ein Datum, wo man die nightly Version genau identifizieren kann.

Hackintool zeigt leider nicht dieses Datum an. Auch ist es offensichtlich, daß Hackintool bei "NVRAM" die Version "unterschlägt".

Die Arbeit mit "git" ist mir nicht fremd. Das läuft schon einige Jährchen. ;-). Was machen die Nichtkundigen?

Das "release wo du zuletzt" mit gearbeitet hast, ist oft nicht mehr bekannt oder auch durch eine "mitgegebene" EFI "verändert" worden.

Wie wollen wir weiter verfahren?