Erledigt El Capitan und die System Integrity Protection - Was ist das und wie kann ich es ändern?

Beitrag von „Griven“ vom 24. September 2015, 22:29

Zusammen mit dem bevorstehenden Release von El Capitan führt Apple auch ein neues Sicherheitsfeature ein welches das System weitestgehend für dem Zugriff von aussen abschotten soll. Natürlich ist hier die Rede von der System Integrity Protection oder, wie das feature anfangs auch genannt wurde, dem Rootless Mode. So ganz neu wie man vielleicht meinen mag ist die SIP aber eigentlich gar nicht denn sie wurde bereits mit OS-X Yosemite eingeführt und von da an konsequent weiterentwickelt. Der allseits bekannte kext-dev-mode ist nämlich bereits ein Teil der SIP und wenn man genauer hinschaut erkennt man, dass der Kernel von Yosemite auch alle anderen Features der SIP unterstützt diese jedoch nicht zum Einsatz kommen. Aber was ist diese SIP denn nun eigentlich genau?

Die System Integrity Protection sichert, wie bereits geschreiben, das System weitestgehend vor dem Zugriff von aussen ab. Dies wird erreicht indem sensible Bereiche des System nicht mehr verändert werden können. Nach der Installation von El Capitan ist die System Integrity Protection standartmäßig aktiviert was zur Folge hat, dass...

-> Diverse Systemverzeichnisse besonders geschützt und ausgeblendet sind (/System, /bin, /sbin, /usr...)
-> Nicht signierte Extensions nicht mehr geladen werden
-> Im Festplattendienstprogramm die Optionen zum reparieren der Berechtigungen fehlen
-> Der Inhalt von Systemverzeichnissen nicht mehr verändert werden kann
-> Der Inhalt des NVRAMs nicht mehr verändert werden kann

Per Design lässt die System Integrity Protection also keinerlei Veränderungen an systemrelevanten Dateien zu und zwar weder durch Benutzer noch durch Programme und selbst der root User ist davon nicht ausgenommen. Für den Otto normalanwender also eigentlich ein durchaus sinnvolles Feature denn das System läuft bei normaler Benutzung mit aktiver SIP nicht anders oder schlechter als ohne nur eben sicherer. Für die Hackintosh Community ist das freilich kein akzeptabler Zustand denn die SIP in der Form in der Apple sie implementiert hat ist für uns eher lästig als nützlich denn sie verhindert zuverlässig, dass unsere Hacks booten da essentielle Extensions wie etwa die FakeSMC jetzt nicht mehr geladen werden. Glücklicherweise lässt sich das Feature sehr fein konfigurien aber dazu muss man erstmal verstehen wie es funktioniert. Die SIP umfasst folgende Restriktionen:

-> Apple Internal
-> Kext Signing
-> Filesystem Protection
-> Debugging Restrictions
-> Dtrace Restrictions
-> NVRAM Protections

und kennt neben an und aus auch noch eine Menge individuelle Konfigurationen die es uns ermöglicht den Schutzgrad unseren Bedürfnissen entsprechend einzustellen. Aber wie funktioniert das Ganze nun? Beim Start des Systems wertet der OS-X eigene Bootloader (boot.efi) bzw. der Kernel den inhalt einer bestimmte NVRAM Variable aus (CSRActiveConfig) und stellt das Schutzlevel entsprechend ihres Inhalts ein. Ist die Variable nicht vorhanden oder enthält den Wert 0 bedeutet dies das die System Integrity Protection komplett aktiv ist, weicht der Wert von der 0 ab wird der Schutzgrad entsprechen der Konfiguration eingestellt. Folgende Übersicht veranschaulicht die möglichen Konfigurationen:

Code

1. hex n/a nvram dtrace intern debug pid fs kexts
2. csrutil enabled --no-internal 00 0 0 0 0 0 0 0 0
3. csrutil enabled 10 0 0 0 1 0 0 0 0
4. csrutil enable —-without kext 11 0 0 0 1 0 0 0 1
5. csrutil enable —-without fs 12 0 0 0 1 0 0 1 0
6. csrutil enable —-without debug 14 0 0 0 1 0 1 0 0
7. csrutil enable —-without dtrace 30 0 0 1 1 0 0 0 0
8. csrutil enable —-without nvram 50 0 1 0 1 0 0 0 0
9. csrutil disabled 77 0 1 1 1 0 1 1 1
10.
11.
12. Other settings
13. disabled (no internal) 67 0 1 1 0 0 1 1 1

Alles anzeigen

Hier wird deutlich, das sich das Schutzlevel sehr fein einstellen lässt. Für einen typischen Hackintosh Fall (Post install) würde es zum Beispiel reichen vorübergehend den Filesystem Schutz auszuschalten und das laden unsignierter Extensions zu erlauben. Folgt man dem oberen Beispiel ergibt sich hieraus also:

Code

1. hex n/a nvram dtrace intern debug pid fs kexts
2. No FileSystem,Kext 03 0 0 0 0 0 0 1 1

Also ein binärwert von 00000011 oder eben ein hex Wert von 03 setzten lässt sich der Wert nun abhängig vom Bootloader auf unterschiedliche Art und Weise. Bei Clover erreicht man das indem man den Wert in den Bereich RT-Variables unter den Punkt CsrActiveConfig in die config.plist einträgt was dann in etwa so aussieht:

Code

1. <key>RtVariables</key>
2. <dict>
3. <key>CsrActiveConfig</key>
4. <string>0x03</string>
5. </dict>

unter Ozmosis gelingt es indem man den Wert mit dem folgenden Kommando in den NVRAM schreibt

Code

1. nvram 7C436110-AB2A-4BBB-A880-FE41995C9F82:csr-active-config=%03

oder an der entsprechenden Stelle in die defaults integriert. Allerdings funktioniert das nicht mehr, wenn El Capitan bereits läuft. Um den Wert zu setzen muss man dann entweder die EIF Shell aus dem Bios starten (HermitShell) und den Wert mittels SetVar Befehl setzen oder aber Yosemite booten und die Einstellung vor der Installation von El Capitan dort vornehmen. Prüfen wie die SIP eingestellt ist kann man dann anschließend im laufenden El Capitan durch die Eingeabe des folgenden Befehls ins Terminal:

Code

1. csrutil status

die Antwort sieht bei komplett abgeschalteter SIP dann so aus

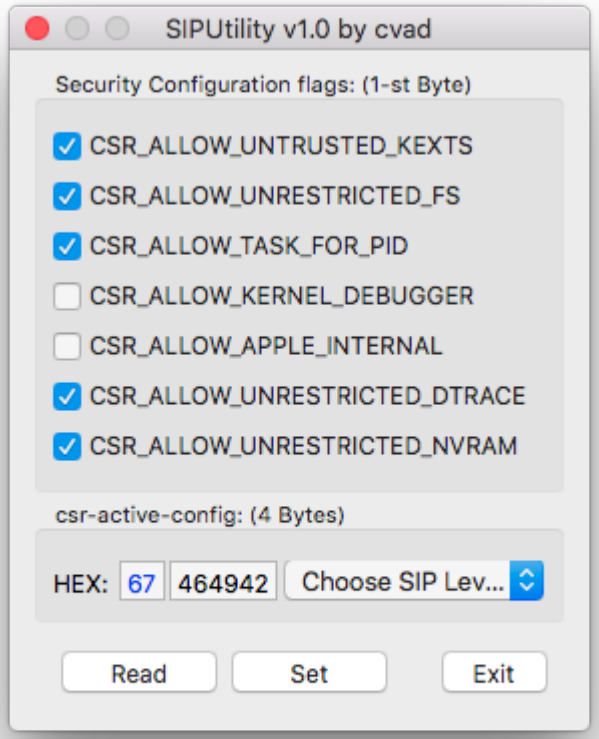
Code

1. System Integrity Protection status: enabled (Custom Configuration).
2.
3.
4. Configuration:

5. Apple Internal: disabled
6. Kext Signing: disabled
7. Filesystem Protections: disabled
8. Debugging Restrictions: disabled
9. DTrace Restrictions: disabled
10. NVRAM Protections: disabled
11.
12.
13. This is an unsupported configuration, likely to break in the future and leave your machine in an unknown state.

Alles anzeigen

Für diejenigen unter Euch, die es lieber grafisch mögen gibt es aber auch ein Tool zum setzen der Einstellungen. Das SIPUtility läuft sowohl auf Yosemite als auch auf El Capitan und lässt komfortable Änderungen an der Konfiguration zu solange der NVRAM beschreibbar ist.



Das Programm habe ich Euch angehangen