System Integrity Protection

Nicolas RUFF – nruff@google.com

Table des matières

1. Présentation

System Integrity Protection[1] (SIP) est un mécanisme de sécurité spécifique à Mac OS X qui bloque l'accès en écriture – même pour l'utilisateur `root` – aux cibles suivantes:

- Les répertoires `/System`, `/bin`, `/sbin` et `/usr` (sauf `/usr/local`).
- Les liens symboliques `/etc`, `/tmp`, `/var`.
- Les processus systèmes signés par Apple. Le blocage inclut l'accès à la mémoire du processus, le débogage et l'analyse avec DTrace.
- Le chargement d'extensions noyau (*kernel extensions* - kext) non signées. Le certificat de signature doit être spécifique à la signature de kexts ; un certificat de développeur AppStore ne suffit pas. Un tel certificat ne coûte que $99 mais ses conditions d'attribution sont discrétionnaires[2].

Le statut actuel de SIP est stocké dans la NVRAM ; il peut être obtenu avec la commande `csrutil`. Toutefois le seul moyen de désactiver cette protection est de passer en mode *Recovery* (+ R au démarrage de la machine), car SIP protège également l'accès à la NVRAM.

```
$ csrutil status
System Integrity Protection status: enabled.

$ sudo touch /System/test
touch: /System/test: Operation not permitted
```

Notons que chaque protection peut être désactivée individuellement (`csrutil enable --without [kext|debug|dtrace|nvram]`).

Il existe également un argument de démarrage du noyau nommé `rootless`, permettant de désactiver cette protection. Mais les arguments de démarrage sont stockés en NVRAM, elle-même protégée par SIP.

L'objectif de cet article est d'analyser la sécurité effective de SIP.

2. Désactiver la protection SIP

SIP a déjà été analysé en détails ici[3] et là[4]. Pour résumer:

---

[1] https://developer.apple.com/library/mac/documentation/Security/Conceptual/System_Integrity_Protection_Guide/Introduction/Introduction.html

[2] https://developer.apple.com/contact/kext

[3] https://derflounder.wordpress.com/2015/10/01/system-integrity-protection-adding-another-layer-to-apples-security-model/

[4] https://reverse.put.as/2015/10/12/rootfool-a-small-tool-to-dynamically-disable-and-enable-sip-in-el-capitan/

- La configuration de SIP est stockée dans `/System/Library/Sandbox/rootless.conf`. Ce fichier détermine les fichiers, répertoires et applications "sensibles". La commande `ls -O` permet de s'assurer si un fichier donné est protégé par SIP (*restricted*).

```
$ ls -lO /
…
drwxr-xr-x+  68 root  wheel  sunlnk            2312 22 oct 09:24 Library
drwxr-xr-x@   2 root  wheel  hidden               68 15 oct 11:25 Network
drwxr-xr-x@   4 root  wheel  restricted          136 20 jan 20:18 System
drwxr-xr-x    6 root  admin  -                   204 15 oct 11:25 Users
…
```

- Une liste d'exceptions (à des fins de rétro-compatibilité) se trouve dans `/System/Library/Sandbox/Compatibility.bundle/Contents/Resources/paths`

La configuration de SIP (activé ou désactivé) est accessible via un appel système dédié (numéro 0x1e3). Il s'agit d'une valeur binaire stockée dans un entier. Il existe une procédure noyau non exportée (`_csr_set_allow_all`) qui permet de désactiver la protection entièrement.

En conclusion, la possibilité d'écrire la valeur zéro à une adresse contrôlée en espace noyau, ou la possibilité d'invoquer `_csr_set_allow_all(1)` en espace noyau, permet de désactiver entièrement la protection SIP.

Note : la distribution stochastique de l'espace d'adressage noyau n'est pas efficace contre l'utilisateur `root`, qui peut obtenir le déplacement relatif via l'appel système `kas_info()`.

Cette attaque est implémentée par les outils RootFool[5] ou kextd_patcher[6], entre autres. Malheureusement il n'existe pas de distribution binaire signée de ces outils, ce qui les rend donc inutilisables en pratique (une extension noyau ne pouvant pas être chargée si elle n'est pas signée).

3. Lacunes d'OS X 10.10

3.1. Désactivation par un argument de démarrage

Avant la version 10.11, le noyau Mac OS X supporte l'argument de ligne de commande `kext-dev-mode` pour autoriser les extensions noyau non signées. Celui-ci est désormais sans effet[7].

3.2. Implémentation de la vérification en espace utilisateur

La vérification de signature présente un défaut énorme: elle est effectuée en espace utilisateur (il s'agit d'un problème connu[8,9]).

Il suffisait donc de recompiler la commande `kextload` depuis les sources publiques, ou de modifier le programme existant, pour contourner la protection. La procédure est la suivante:

1. Modifier la fonction `checkKextSignature` dans la commande `kextload`. Ne pas oublier d'enlever la signature de l'application[10], qui est invalidée par l'opération.
2. Arrêter le service `kextd`, pour obliger la commande `kextload` à effectuer le chargement elle-même.

```
# launchctl unload /System/Library/LaunchDaemons/com.apple.kextd.plist
```

3. Profiter ?

```
# kextload.patched.unsigned -v rootfool.kext

Can't contact kextd; attempting to load directly into kernel.
```

[5] https://github.com/gdbinit/rootfool
[6] https://github.com/Tyilo/kextd_patcher
[7] https://developer.apple.com/library/mac/documentation/Security/Conceptual/System_Integrity_Protection_Guide/KernelExtensions/KernelExtensions.html
[8] https://reverse.put.as/2013/11/23/breaking-os-x-signed-kernel-extensions-with-a-nop/
[9] https://www.blackhat.com/docs/us-15/materials/us-15-Wardle-Writing-Bad-A-Malware-For-OS-X.pdf
[10] https://github.com/steakknife/unsign

```
Loading .../rootfool.kext.

kext-dev-mode allowing invalid signature -67050 0xFFFFFFFFFFFFEFA16 for kext
'.../rootfool.kext'
kext signature failure override allowing invalid signature -67050 0xFFFFFFFFFFFFEFA16 for kext
".../rootfool.kext"

(kernel) Not entitled to link kext 'com.sentinelone.rootfool'
(kernel) Failed to load executable for kext com.sentinelone.rootfool.
(kernel) Kext com.sentinelone.rootfool failed to load (0xdc008004).
(kernel) Failed to load kext com.sentinelone.rootfool (error 0xdc008004).
Failed to load ./rootfool.kext - (libkern/kext) not privileged.
.../rootfool.kext failed to load - (libkern/kext) not privileged.
```

Comme on le voit dans la capture ci-dessus, cette attaque ne fonctionne plus sur Mac OS X version 10.11.

Le problème vient du fait que la commande `kextload` doit être signée par Apple et posséder un *entitlement* spécifique pour pouvoir charger une extension du noyau – précisément à cause de SIP. Une commande recompilée ou modifiée ne sera pas reconnue, comme l'atteste cet extrait de `libkern/c++/OSKext.cpp`:

```
/* <rdar://problem/21444003> all callers must be entitled */

if (FALSE == IOTaskHasEntitlement(current_task(), "com.apple.rootless.kext-management")) {
  kOSKextLogErrorLevel | kOSKextLogLoadFlag,
  "Not entitled to link kext '%s'",
  result = kOSKextReturnNotPrivileged;
}
```

Note: il est possible d'auditer le chargement d'une extension noyau via les commandes suivantes:

- `spctl --assess -vv` (donne un résultat binaire basé sur la politique de sécurité courante).
- `kextutil`
- Eventuellement, la commande `codesign -vvv`

On pourrait envisager de contourner cette protection en modifiant la commande `kextload` en mémoire plutôt que sur disque, mais SIP empêche également le débogage du processus `kextload`.

3.3. Liste d'exceptions

L'extension `/System/Library/Extensions/AppleKextExcludeList.kext` contient les ressources suivantes :

- `./Contents/Info.plist`
  - `OSKextExcludeList` (actuellement 29 entrées, identifiées par leur nom et leur version).
  - `OSKextSigExceptionHashList` (actuellement 11726 entrées, identifiées par leur nom, leur version et leur condensat SHA1).
- `./Resources/KnownPanics.plist`

La première liste (`OSKextExcludeList`) contient une liste noire d'extensions. Il existe déjà une liste d'extensions incompatibles (`KnownPanics.plist`), cette première liste est donc une liste d'extensions spécifiquement malveillantes ou compromises.

L'une de ces extensions est facile à trouver ; il s'agit de DockMod4[11]. Cette extension permet de tester l'efficacité de la liste noire :

```
$ sudo kextutil DockmodDriver.kext
DockmodDriver.kext is in exclude list; omitting.
```

L'autre liste contient une liste d'extensions non signées mais autorisées. Un condensat SHA1 est utilisé pour identifier l'extension ; l'algorithme permettant de calculer ce condensat sur l'ensemble du contenu de l'extension se trouve dans `getAdhocSignatureHash()`[12].

[11] https://www.spyresoft.com/dockmod
[12] https://opensource.apple.com/source/kext_tools/kext_tools-426.20.2/security.c

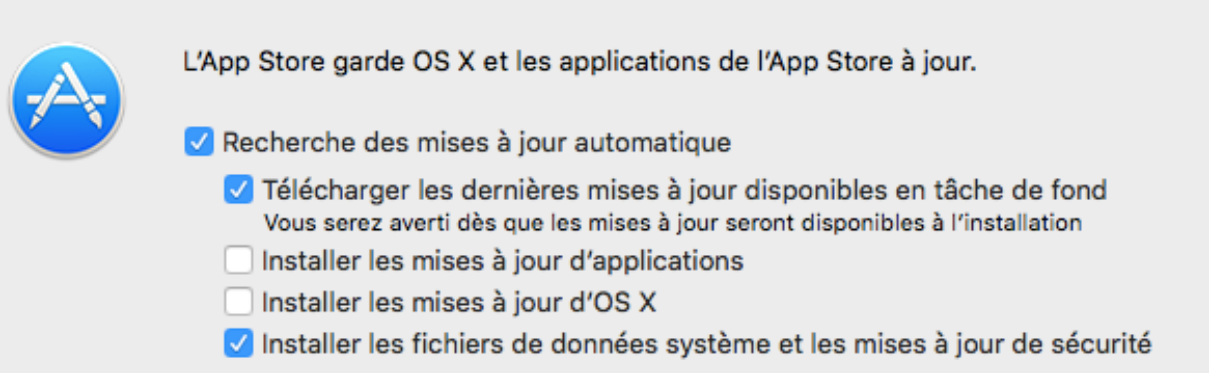
Toutes les versions de l'outil SAT SMART Driver[13] se trouvent dans cette liste blanche, ce qui permet de tester son fonctionnement. On constate que Mac OS X 10.10 accepte effectivement le chargement d'une extension non signée en liste blanche. A contrario, la commande `kextload` dans Mac OS 10.11 honore la politique de sécurité SIP :

```
/*********************************************************************
 * isInvalidSignatureAllowed() - check if kext with invalid signature is
 * allowed to load.  Currently we check to see if we are running with boot-args
 * including "kext-dev-mode".  In the future this is likely be removed or
 * changed to use other methods to set up machines in "developer mode".
 *********************************************************************/
Boolean isInvalidSignatureAllowed(void)
{
    Boolean      result = false;      // default to not allowed

    if (csr_check(CSR_ALLOW_UNTRUSTED_KEXTS) == 0 || csr_check(CSR_ALLOW_APPLE_INTERNAL) == 0) {
        // Allow kext signature check errors
        result = true;
    }
    else {
        // Do not allow kext signature check errors
        OSKextLog(/* kext */ NULL,
                  kOSKextLogErrorLevel  | kOSKextLogGeneralFlag,
                  "Untrusted kexts are not allowed");
    }

    return(result);
}
```

Note : ces listes (noires et blanches) peuvent être mise à jour de manière silencieuse dans la configuration par défaut du système. C'est ainsi que le pilote Ethernet a « accidentellement » été désactivé en février 2016 par une mise à jour[14].



3.4. Autres attaques connues

Il existe d'autres approches pour autoriser le chargement d'extensions non signées "officiellement" ; en effet il s'agit d'une opération nécessaire à l'installation de Mac OS X sur du matériel non-Apple (ex. `FakeSMC.kext`). Ce domaine faisait donc l'objet d'une recherche intense dans le milieu du Hackintosh[15].

L'attaque GateBreak[16] ajoute une autorité de certification valide pour la signature d'extensions. Cette attaque modifie les commandes `kext*` et ne fonctionne que sur OS X versions 10.9 et 10.10.

Aujourd'hui le contournement de signature n'est plus à la mode, et les attaquants proposent désormais des chargeurs d'amorçage EFI (tels que Clover[17] ou Chameleon[18]).

[13] https://github.com/kasbert/OS-X-SAT-SMART-Driver/releases
[14] http://arstechnica.com/apple/2016/02/os-x-blacklist-accidentally-disables-ethernet-in-os-x-10-11/
[15] http://www.hackintosh.com/
[16] http://www.tonymacx86.com/mavericks-desktop-support/112306-gatebreak-signed-kexts-everyone.html
[17] https://sourceforge.net/projects/cloverefiboot/
[18] http://chameleon.osx86.hu/

4. Conclusion

Avant Mac OS X 10.11, la sécurité offerte par SIP est trivialement contournable par de multiples méthodes.

Mac OS X 10.11 comble ces failles par l'identification des composants « de confiance » au travers d'une signature spécifique (*entitlement*), et la suppression des mécanismes de rétro-compatibilité. Cette approche renforce le contrôle d'Apple sur l'écosystème OS X – il est désormais difficile de proposer une alternative à la commande `kextload`. Le développement d'extensions noyau par des projets Open Source est soumis à la délivrance discrétionnaire d'un certificat par Apple.

A ce jour il n'existe que 3 CVE liés aux commandes `kext*`, ce qui est probablement trop peu, compte-tenu de la complexité du système :

- CVE-2015-3708[19] permet d'abuser le chargeur d'extensions grâce à un lien symbolique sous OS X < 10.10.4
- CVE-2015-3709[20] est une concurrence temporelle affectant OS X < 10.10.4
- CVE-2015-7052[21] est une faille non spécifiée affectant OS X < 10.11.2

Il est donc encore un peu tôt pour déclarer le chargement d'extensions noyau « sûr ».

Par ailleurs la désactivation du contrôle de signature devient un vecteur d'exploitation noyau intéressant dans le cadre d'élévations de privilèges `root` vers noyau, que SIP tente de prévenir. C'est un vecteur d'exploitation utilisé récemment par Google Project Zero[22] et probablement destiné à susciter un intérêt croissant dans la communauté des chercheurs en sécurité OS X.

[19] https://support.apple.com/en-us/HT204942 ; https://bugs.chromium.org/p/project-zero/issues/detail?id=343
[20] https://support.apple.com/en-us/HT204942 ; https://bugs.chromium.org/p/project-zero/issues/detail?id=353
[21] https://support.apple.com/en-us/HT205637
[22] https://googleprojectzero.blogspot.com/2016/03/race-you-to-kernel.html