MDM and the Kextpocalypse 2 – Richard Purves

Published: 2017-11-10 · Archived: 2026-04-05 17:15:31 UTC

Yes there's another Kextpocalypse heading the Mac Admin's way. Unusually it's going to hit on the next point release of macOS. 10.13.2 is going to have a LOT to answer for, as is Apple.

Before we continue, I need to reference my sources here as this info isn't officially public but it's discussed in enough public forums not to matter. First up is an open radar called "Please document the new 10.13.2 User-Approved MDM Enrollment and new MDM payloads outside of AppleSeed"

which I highly suggest you read first.

Next up is a public Apple Technical Note, number TN2459 to be exact which provides some other information. Take your time, read both and once you're done come back to me.

..

What the hell Apple? You can't just introduce wide sweeping technical changes like that with almost ZERO notice and what notice there was is buried behind the Appleseed program which is **INVITE ONLY** . I mean, if this hadn't got out there'd be so many highly annoyed admins out there scrambling to unbreak what you're about to break.

More warning in future please Apple. Ok done. On with the rest of the post. The salient points are as follows.

- There is now "User Approved Mobile Device Management" or UAMDM.
- There is now "User Approved Kernel Extension Loading" or UAKEL.
- If your device is DEP enrolled into an MDM solution, you don't need to worry about UAMDM but UAKEL is still a concern.
- If your device isn't DEP enrolled into an MDM, you will have restricted management options unless the user grants them to you.
- UAKEL is disabled if you have an MDM *for now*.
- UAKEL can have a whitelist via a configuration profile.

Oh and all you non MDM managed devices are utterly screwed if you want to deploy software with kext's in it. Non MDM solutions just got a lot nastier to use.

After LOTS of chat today in the Mac Admin Slack, especially in #security it can all be summed up as follows.

DEP enrolled into an MDM, with a profile to whitelist kexts or game over.

This is an enterprise nightmare and I don't say that lightly. I deal with high security type environments where all the stuff the users would approve kexts and apps is *disabled by design* and also *by security requirement*. This is really going to make life difficult. Again, why is this in a point release? You may have enough time to prepare if you all start now.

I can't do much about DEP and I think this is another nail in the coffin of imaging macs, but remember I mentioned that you can whitelist kexts in a profile. That I can help you with, and I've done a lot of work today trying to decipher what limited information is available to me.

com.apple.syspolicy.kernel-extension-policy is the new payloadtype that must be delivered via a User Approved MDM. (See? Can't even whitelist the apps you might need unless user says yes or you DEP'd things.) It has three keys that are quite important. Let's go through them.

- AllowUserOverrides – This controls is a user is allowed to approve kexts for use that aren't in the profile
- AllowedTeamIdentifiers – This is an approved list of "team identifiers", so that their kexts will be allowed to load.
- AllowedKernelExtensions – This is a map of "team identifiers" to "bundle identifiers". You can basically specify what kexts for a given vendor you want to load instead of all of them.

A "Team Identifier" is a string that looks like this: "DX1G23M4N5" and appears to be tied to the vendor's Apple ID. It appears unique to each vendor.

A "Bundle Identifier" is part of the Info.plist inside each kext bundle. It's actually the CFBundleIdentifier field and anyone who's done app packaging should be familiar with it as it's a unique identifier for that kext.

But how does that work in a profile? Let's take a look at one i've managed to piece together from the scraps of info out there.

[cc lang="xml"]

AllowUserOverrides

AllowedTeamIdentifiers

team id here
team id here

AllowedKernelExtensions

team id

bundle id here

unsigned bundle id here

[/cc]

I've annotated the example to give you an idea of what goes where. The AllowUserOverrides is a boolean true/false condition. The AllowedTeamIdentifiers is an array list of the team id's and the AllowedKernelExtensions is a key for the team id then an array of every kext approved made by that team id.

You wouldn't, for example, put an id in the AllowedTeamIdentifiers then specify the same id with specific kext bundle ids in AllowedKernelExtensions. You would mix and match depending on whether you want to whitelist globally or for specific kext files.

~~The exception is if you're a developer with an unsigned kext under test, you can leave the key blank and just specify the bundle identifier on it's own. Note, you won't be able to distribute your kext by Apple without having it signed first!~~

Since posting I've been informed that the above isn't the case. Unsigned kexts during development cannot be loaded on a normal security-enabled system, so the profile cannot allow them to load. Whoops.

The really hard part is working out what this data is in the first place! I've already mentioned how you can get the CFBundleIdentifier from inside the kext but how do you work out what the Team Identifier is? Well turns out that "codesign" in terminal is our friend. Using MalwareBytes installed kext as an example .. behold!

[cc lang="bash"]
codesign -d -vvvv /Library/Extensions/com.malwarebytes.mbam.rtprotection.kext 2>&1 | grep "Authority=Developer ID Application:" | cut -d"(" -f2 | tr -d ")"
GVZRY6KDKR
[/cc]

So we can now work out what the Team Identity for a given kext is, we can find the Bundle Identity for a given kext and we've an idea how to configure the whitelist plist … how do you find these things? They can be scattered all over the place. The answer lies in the following script that I quickly through together today.

[cc lang="bash"]
#!/bin/bash

Script to scan a system for kexts and gather the information needed for Apple whitelisting
richard at richard – purves dot com

# Stop IFS linesplitting on spaces
OIFS=$IFS
IFS=$'\n'

Scan the following folders to find 3rd party kexts
/Applications
/Library/Extensions
/Library/Application Support

```
echo “Searching Applications folder”
applic=($( find /Applications -name “*.kext” ))

echo “Searching Library Extensions folder”
libext=($( find /Library/Extensions -name “*.kext” -maxdepth 1 ))

echo “Searching Library Application Support folder”
libapp=($( find /Library/Application\ Support -name “*.kext” ))

echo “”

# Merge the arrays together
results=(“${applic[@]}” “${libext[@]}” “${libapp[@]}”)

if [ ${#results[@]} != “0” ];
then
for (( loop=0; loop<${#results[@]}; loop++ )) do # Get the Team Identifier for the kext teamid=$( codesign -d -vvvv ${results[$loop]} 2>&1 | grep “Authority=Developer ID Application:” | cut -d”(” -f2 | tr -d “)” )

# Get the CFBundleIdentifier for the kext
bundid=$( defaults read “${results[$loop]}”/Contents/Info.plist CFBundleIdentifier )

echo “Team ID: $teamid Bundle ID: $bundid”
done
fi

IFS=$OIFS

exit
```

Run with sudo, this’ll give an output similar to this:

```
Richards-MacBook-Pro:Desktop richardpurves$ sudo ./findkextinfo.sh
Password:
Searching Applications folder
Searching Library Extensions folder
Searching Library Application Support folder

Team ID: EG7KH642X6 Bundle ID: com.vmware.kext.vmnet
Team ID: EG7KH642X6 Bundle ID: com.vmware.kext.vmci
Team ID: EG7KH642X6 Bundle ID: com.vmware.kext.vmx86
Team ID: EG7KH642X6 Bundle ID: com.vmware.kext.vmioplug.17.1.2
Team ID: FC94733TZD Bundle ID: com.ATTO.driver.ATTOExpressSASHBA2
Team ID: K3TDMD9Y6B Bundle ID: com.Accusys.driver.Acxxx
Team ID: NDGSU3WA4Y Bundle ID: com.softraid.driver.SoftRAID
```

Team ID: DX6G69M9N2 Bundle ID: com.highpoint-tech.kext.HighPointIOP
Team ID: 8R7PS6VYW7 Bundle ID: com.CalDigit.driver.HDPro
Team ID: DX6G69M9N2 Bundle ID: com.highpoint-tech.kext.HighPointRR
Team ID: 34JN824YNC Bundle ID: com.Areca.ArcMSR
Team ID: FC94733TZD Bundle ID: com.ATTO.driver.ATTOCelerityFC8
Team ID: 6HB5Y2QTA3 Bundle ID: com.hp.kext.io.enabler.compound
Team ID: 268CCUR4WN Bundle ID: com.promise.driver.stex
Team ID: FC94733TZD Bundle ID: com.ATTO.driver.ATTOExpressSASRAID2
Team ID: GVZRY6KDKR Bundle ID: com.malwarebytes.mbam.rtprotection
[/cc]

Run this script with sudo, so it can work it's way past any permissions issues. McAfee likes to lock it's own kexts down so nothing can tamper with them, for example. It'll produce an on screen list that'll make generating your own kext whitelist a lot easier than doing it all by hand.

Add to your MDM, and deploy in the usual way via APNS. That'll help but remember the big lesson i've drawn from this is DEP is now the official way forward for enterprises and we should embrace it very quickly.
