PwnBoot by the TK_Cracks Team

This writeup will detail the inner workings of the PwnBoot jailbreak for iOS 6 by the TK_Cracks jailbreak team.

Part 1: Intro:

- **What is a Jailbreak?**
    - Per TheiPhoneWiki, a jailbreak is “the process by which full execute and write access is obtained on all the partitions of the Apple TV, iPad, iPad mini, iPhone or iPod touch.” Most jailbreaks install a package manager called Cydia, which assists in installing tweaks to the device in the form of .deb files.
- **What goes into a Jailbreak?**
    - An acceptable barebones jailbreak consists of modifying `/etc/fstab` to mount the System partition as read-write, and in addition installing the AFC2 service, a modified AFC service which has access to the whole filesystem.
- **Key Terms**:
    - Jailbreak: (see above)
    - `/etc/fstab`: The file that controls read/write access of the root and media partitions
    - AFC/AFC2: Service which iTunes uses to exchange files with a device

---

Objective:

Via a custom ramdisk, achieve read/write access on the root partition of an iOS device, along with installing the AFC2 service, installing an ssh service, patching the kernel (sandbox,codesign, etc.), and installing Cydia

Research / Vulnerabilities

For this jailbreak, we have to find a way to load a custom ramdisk onto our device, as this is the whole purpose and method of this jailbreak. Luckily, the limera1n BootROM exploit allows us to load patched IMG3s! With Limera1n, we can send a custom ramdisk (after sending other necessary components), using a tool called `irecovery` by westbaer.

Procedure

1. Patch iBSS and iBEC
2. Build a Ramdisk with a SSH service set to run at boot, allowing us access to the full RootFS of our device
3. Patch kernelcache to nuke AMFI, allowing us to run said SSH service along with any other unsigned binaries we need.
4. Enter pwned DFU mode via limera1n
5. Send components in appropriate order, boot ramdisk

6. SSH into device over USB
7. Mount RootFS partition
8. Patch `fstab` and install `AFC2`
9. (ADVANCED: For a full jailbreak, abuse `launchd.conf` to start p0sixspwn on boot)

THE PROCESS

Part 2: Preparing and Booting a Custom Ramdisk

Before we start, we must download all the necessary bootchain components from Apple:

```
partialzip "http://appldnld.apple.com/iOS5.1.1/041-
4347.20120427.o2yov/iPhone2,1_5.1.1_9B206_Restore.ipsw"
"Firmware/dfu/iBEC.n88ap.RELEASE.dfu" "ibec.dfu"
partialzip "http://appldnld.apple.com/iOS5.1.1/041-
4347.20120427.o2yov/iPhone2,1_5.1.1_9B206_Restore.ipsw"
"Firmware/dfu/iBSS.n88ap.RELEASE.dfu" "ibss.dfu"
partialzip "http://appldnld.apple.com/iOS5.1.1/041-
4347.20120427.o2yov/iPhone2,1_5.1.1_9B206_Restore.ipsw" "kernelcache.release.n88"
"kern.n88"
partialzip "http://appldnld.apple.com/iOS5.1.1/041-
4347.20120427.o2yov/iPhone2,1_5.1.1_9B206_Restore.ipsw"
"Firmware/all_flash/all_flash.n88ap.production/DeviceTree.n88ap.img3"
"devicetree.img3"
partialzip "http://appldnld.apple.com/iOS5.1.1/041-
4347.20120427.o2yov/iPhone2,1_5.1.1_9B206_Restore.ipsw" "038-4349-020.dmg"
"ramdisk.dmg"
```

Next, we have to patch iBSS and iBEC:

```
xpwntool ibss.dfu ibss.dfu.dec -iv 0cbb6ea94192ba4c4f215d3f503279f6 -k
36782ee3df23e999ffa955a0f0e0872aa519918a256a67799973b067d1b4f5e0
fuzzy_patcher --patch --orig ibss.dfu.dec --patched ibss.dfu.dec.p --delta ibss.patch
move ibss.dfu ibss.dfu.orig
xpwntool ibss.dfu.dec.p ibss.dfu -t ibss.dfu.orig -iv 0cbb6ea94192ba4c4f215d3f503279f6
-k 36782ee3df23e999ffa955a0f0e0872aa519918a256a67799973b067d1b4f5e0

xpwntool ibec.dfu ibec.dfu.dec -iv 1fe15472e85b169cd226ce18fe6de524 -k
677be330d799ffafad651b3edcb34eb787c2d6c56c07e6bb60a753eb127ffa75
fuzzy_patcher --patch --orig ibec.dfu.dec --patched ibec.dfu.dec.p --delta
ibec.verbose.patch
move ibec.dfu ibec.dfu.orig
xpwntool ibec.dfu.dec.p ibec.dfu -t ibec.dfu.orig -iv 1fe15472e85b169cd226ce18fe6de524
-k 677be330d799ffafad651b3edcb34eb787c2d6c56c07e6bb60a753eb127ffa75
```

Now, somewhere along the line here, we are going to need to run unsigned binaries, like the SSH service, to do the post-boot work. In order to do so, we must patch the kernelcache and nuke AMFI:

```
xpwntool kern.n88 kern.n88.dec -iv 0dc795a64cb411c21033f97bceb96546 -k
0cc1dcb2c811c037d6647225ec48f5f19e14f2068122e8c03255ffe1da25dec3
fuzzy_patcher --patch --orig kern.n88.dec --patched kern.n88.dec.p --delta
kern.n88.patch
move kern.n88 kern.n88.orig
xpwntool kern.n88.dec.p kern.n88 -t kern.n88.orig -iv 0dc795a64cb411c21033f97bceb96546
-k 0cc1dcb2c811c037d6647225ec48f5f19e14f2068122e8c03255ffe1da25dec3
```

Finally, we must build the custom ramdisk itself! The `ssh.tar` file includes a bundled ssh service scheduled to run at boot, and any other necessary binaries.

```
xpwntool ramdisk.dmg ramdisk.dmg.dec -iv 26ec90f47073acaa0826c55bdeddf4bb -k
7af575ca159ba58b852dfe1c6f30c68220a7a94be47ef319ce4f46ba568b7a81
hfsplus ramdisk.dmg.dec grow 45000000
hfsplus ramdisk.dmg.dec untar ssh.tar "/"
move ramdisk.dmg ramdisk.dmg.orig
xpwntool ramdisk.dmg.dec ramdisk.dmg -t ramdisk.dmg.orig -k
7af575ca159ba58b852dfe1c6f30c68220a7a94be47ef319ce4f46ba568b7a81 -iv
26ec90f47073acaa0826c55bdeddf4bb
```

Now, all of the required components are built and patched!

Next up is actually booting them!

Part 3: Booting patched/custom components

Before doing this part, you must enter your device into pwned dfu mode. You can do this via sn0wbreeze or irecovery, although i prefer irecovery, since it is way faster.

```
irecovery -f "ibss.dfu"
timeout /t 5 /nobreak > NUL
echo Sending iBEC
irecovery -f "ibec.dfu"
timeout /t 5 /nobreak > NUL
```

(Wait for the device to enter recovery mode [signaled by the backlight turning on] and then proceed:)

```
timeout /t 5 /nobreak > NUL
echo Sending DeviceTree
irecovery -f "devicetree.img3"
irecovery -c devicetree
timeout /t 5 /nobreak > NUL
echo Sending Ramdisk
irecovery -f "ramdisk.dmg"
irecovery -c ramdisk 0x90000000
```

```
timeout /t 5 /nobreak > NUL
echo Sending Kernelcache
irecovery -f "kern.n88
timeout /t 5 /nobreak > NUL
echo Booting kernelcache
irecovery -c bootx
```

Your device should now be showing an Apple logo with a blank progress bar, signaling that the SSH service is halting boot. Now you have to forward the SSH service over USB (port 2022):

```
itunnel_mux --lport 2022
```

Now you are able to SSH into the device over USB. Leave the previous window open, and open a new terminal/CMD window to do so (`root@127.0.0.1` over port `2022` with password `alpine`).

Now that you've successfully SSH'd into the device, we must mount the root filesystem to gain access to all our files. In the window where you SSH'd into the device, run `mount.sh` which should report something like "`mounting /mnt1`". Now we proceed to the post-boot work.

Part 3: Post-boot work

Patch `/mnt1/etc/fstab` for R/W on the RootFS:

Pre-patch:

```
/dev/disk0s1s1 / hfs ro 0 1
/dev/disk0s1s2 /private/var hfs,nosuid,nodev rw 0 2
```

Post-patch:

```
/dev/disk0s1s1 / hfs rw 0 1
/dev/disk0s1s2 /private/var hfs,nosuid,nodev rw 0 2
```

Install AFC2 for easy file transfer:

This involves `cp`ing the `Services.plist` file located in `/bin` to `/mnt1/System/Library/Lockdown/Services.plist`, overwriting the default one that should be there.

Make other RootFS modifications

At this point, feel free to make any other RootFS modifications you'd like. This could include deleting files, pre-installing games, or even installing p0sixspwn and Cydia for a fully-untethered jailbreak! The possibilities are basically endless! Once you're done, run either `reboot` or `kill` to restart your iPhone and have it boot back into iOS!

Conclusions:

There we go! We have successfully barebones jailbroken an iPhone using a custom ramdisk! As shown, this method is fairly simple and very friendly to beginner devs. If you have any questions about this procedure, or want to know more about its potential applications, feel free to send me a DM on Twitter (@iBoot32) and I'll respond as soon as I can! Also, follow my jailbreak team @TK_Cracks for more writeups and tools like this!

~ TK_Cracks Team