IOS Cracking

From Hackulous Wiki
Jump to: navigation, search

iOS cracking is the process by which iOS applications are decrypted (cracked) so they may be used on other jailbroken devices. The method used is crude but simple: a debugger is attached to the executable and is used to dump the decrypted segments before the executable launches. The decrypted segments are then transposed onto the original binary, and the `LC_ENCRYPTION_INFO` load command's `cryptid` field is changed to 0.

Contents

History

After the release of the App Store, lsemtex, Iceman-fr and cdecl gathered in an IRC channel (#ipa) and investigated the way Apple applied FairPlay DRM to iOS applications. They ported gdb to the iPhone and devised a process by which the

Before long, Flox created the first automated cracking script, autop. Later xCrack became the first utility to package the cracked application into an IPA file.

Today, iOS cracking is automated with Clutch and poedCrackMod.

Basics

Note: This section explains how Clutch cracks iOS applications, because the method Clutch uses is similar to the way that GDB (and most debuggers) work internally.

Application Analysis

iOS applications are installed into the `/var/mobile/Applications/` directory within a randomly named payload directory. This directory will contain the `.app` directory (application data), the `iTunesMetadata.plist` dictionary file (containing some sensitive information about the purchaser), and the `Documents`, `Library` and `tmp` directories.

The executable is located within the `.app` directory, and is always named by the `CFBundleExecutable` key within its accompanying `.app/Info.plist` dictionary file. Some parts of this executable will be encrypted if the application has been purchased from the App Store. To check if an application is encrypted, use the `otool -l` command:

```
# otool -l iSilo | grep LC_ENCRYPTION_INFO -A 4
          cmd LC_ENCRYPTION_INFO
      cmdsize 20
 cryptoff  4096
 cryptsize 1347584
 cryptid   1
```

In the above output, `cryptid` is 1, meaning the application is encrypted. After being cracked (decrypted), `cryptid` is set to 0 to prevent the kernel from trying to decrypt already-decrypted data.

Within the `.app` directory there is also an SC_Info directory which contains keys used to decrypt the executable. These keys are used by fairplay in conjunction with the iTunes library key list and device's MAC Address and other identifiers. The `SC_Info` directory's contents are sensitive and specific to the purchaser, and must be removed or corrupted before distribution.

Executable files on iOS are Mach-O files (also used on Mac OS X), and are documented here. Some executables are fat binaries, meaning they contain multiple mach objects within a single file, each one for a different architecture or platform.

Typical format of an iOS Mach-O file

To detect if an executable is a fat binary, use the `otool -f` command like so:

```
# otool -f iSilo
Fat headers
fat_magic 0xcafebabe
nfat_arch 2
architecture 0
    cputype 12
    cpusubtype 6
    capabilities 0x0
    offset 4096
    size 1488304
    align 2^12 (4096)
architecture 1
    cputype 12
    cpusubtype 9
    capabilities 0x0
    offset 1495040
    size 1495376
    align 2^12 (4096)
```

As you can see, the executable `iSilo` has two architectures. On iOS devices (which use the ARM processor instruction set) cpusubtype 6 means ARMV6, and cpusubtype 9 means ARMV7. The mach loader will choose the best architecture to match the device (newer devices run ARMV7). ARMV6 devices cannot execute ARMV7 architectures, so on ARMV6 devices the fat binary is usually "thinned" into an ARMV6 binary before cracking begins.

In a fat binary, the `fat_header` and subsequent `fat_arch` array are identified using a binary magic (0xcafebabe). Padding is added to round the file to the nearest memory page (0x1000) and the first mach object usually starts at 0x1000 from the start of the file. Files which are thin (only one Mach object) are identified with the mach-o binary magic (0xdeadbeef).

Applications are decrypted by the kernel before the executable is launched. The mach loader identifies the `LC_ENCRYPTION_INFO` load command and uses the keys within `SC_Info` (along with other iTunes/device identifiers) to decrypt the segment after it has been loaded in memory. These keys are usually cached by the loader (or the fairplay decryption agent), meaning that cracking both architectures efficiently may require moving these keys and changing the executable's filename. (This is performed by Clutch.)

reflect that the executable has been cracked.

All changes to a Mach object must be reflected within the CodeSignature hash table, located within the __LINKEDIT segment. This can be done automatically with the ldone utility.

Using GDB to Dump

The executable's decrypted segment can be dumped with GDB using a GDB batch script like so:

```
$CryptSize=1347584
$CryptOff=4096
echo -e "set sharedlibrary load-rules \".*\" \".*\" none\r\n\
set inferior-auto-start-dyld off\r\n\
set sharedlibrary preload-libraries off\r\n\
set sharedlibrary load-dyld-symbols off\r\n\
dump memory dump.bin $(($CryptOff + 4096)) $(($CryptSize + $CryptOff + 4096))\r\n\
kill\r\n\
quit\r\n" > batch.gdb

gdb -q -e iSilo -x batch.gdb -batch
```

This method will dump the architecture chosen by the mach loader (the one most appropriate for your device). To dump the other architecture, you will have to change the executable's name (and the `SC_Info` key names) and swap the ARMV6 and ARMV7 `cpusubtypes`.

Defeating ASLR

ASLR can be defeated in several ways. The MH_PIE flag within the mach_header can simply be unset before a debugger is used to dump the data, and then set after the data has been dumped. This method, however, requires resigning the binary. posix_spawn can be provided a spawn flag of _POSIX_SPAWN_DISABLE_ASLR (0x0100) to disable ASLR.

Clutch uses `vm_region` to identify the starting (non __PAGEZERO) region for the image, thus determining the `vmaddr` slide.

Packaging an IPA

See also: IPA

Cracked iOS applications are packaged into the IPA format, which is also used by iTunes to manage legitimate applications. This format has a unique structure:

- `Payload/` contains the .app directory for the application. **Remember: the SC_Info directory within the .app directory must be removed or censored, as it contains sensitive keys.**
- `iTunesArtwork` is a 512x512 icon of the application, used by iTunes.
- `iTunesMetadata.plist` (optional) contains iTunesMetadata used by the App Store app and iTunes to check for updates. This file does not need to be removed, but if it remains several fields (`appleId` and `purchaseDate`) must be censored.

Directories located within an installed version of the application (`Documents`, `tmp`, or `Library`) are not included within an IPA file.

Retrieved from "http://hackulo.us/wiki/IOS_Cracking"

- Discussion
- View source
- History