Jailbreak Scanner for IOS

Jailbreak Scanner provides a comprehensive way to detect **jailbroken iOS** devices. The plugin uses multiple detection methods to reliably identify jailbroken environments, helping developers implement appropriate security measures for their applications.

Features

JailbreakScanner utilizes multiple detection techniques to identify jailbroken devices and prevent unauthorized modifications. Below is a breakdown of each detection method:

1. File System Checks

Identifies the presence of known jailbreak-related files and directories that should not exist on a standard iOS device. The scanner verifies:

- Existence of common jailbreak tools such as:
  - `/Applications/Cydia.app`
  - `/Applications/Sileo.app`
  - `/Applications/Zebra.app`
  - `/Applications/FakeCarrier.app`
  - `/Applications/WinterBoard.app`
  - `/Applications/SBSettings.app`
  - `/Applications/RockApp.app`
  - `/Applications/blackra1n.app`
  - `/Applications/MxTube.app`
  - `/Applications/IntelliScreen.app`
  - `/usr/sbin/frida-server`
  - `/usr/bin/ssh`
  - `/usr/sbin/sshd`
  - `/usr/libexec/ssh-keysign`
  - `/usr/lib/libhooker.dylib`
  - `/usr/lib/libjailbreak.dylib`
  - `/usr/lib/libsubstitute.dylib`
  - `/usr/lib/substrate`
  - `/usr/lib/TweakInject`
- Presence of package managers or third-party repositories:
  - `/etc/apt/sources.list.d/cydia.list`
  - `/etc/apt/sources.list.d/sileo.sources`
  - `/etc/apt/sources.list.d/electra.list`
  - `/var/lib/dpkg/info/mobilesubstrate.md5sums`
  - `/Library/MobileSubstrate/MobileSubstrate.dylib`
  - `/Library/MobileSubstrate/DynamicLibraries/`
  - `/Library/MobileSubstrate/DynamicLibraries/LiveClock.plist`

    - `/Library/MobileSubstrate/DynamicLibraries/Veency.plist`
- Existence of jailbreak-related configuration files:
    - `/.bootstrapped_electra`
    - `/.installed_unc0ver`
    - `/jb/offsets.plist`
    - `/jb/jailbreakd.plist`
    - `/jb/amfid_payload.dylib`
    - `/jb/libjailbreak.dylib`
    - `/var/mobile/Library/SBSettings/Themes`
    - `/private/var/tmp/cydia.log`
    - `/private/var/lib/apt/`
    - `/private/var/lib/cydia`
    - `/private/var/cache/apt/`
    - `/private/var/log/syslog`
    - `/System/Library/LaunchDaemons/com.ikey.bbot.plist`
    - `/System/Library/LaunchDaemons/com.saurik.Cydia.Startup.plist`

If any of these files are found, the device is likely jailbroken.

2. Inspection of Suspicious Loaded Libraries

Detects unauthorized or injected libraries that indicate tampering with system functionality. The scanner checks loaded dynamic libraries (`DYLD` hooks) for:

- `MobileSubstrate.dylib` (Cydia Substrate used for tweak injection)
- `libhooker.dylib` (used in jailbreak frameworks like Odyssey and Taurine)
- `SubstrateLoader.dylib`, `SSLKillSwitch2.dylib`, and `TweakInject.dylib` (common hacking tools)
- Any Frida-based debugging or hooking frameworks

3. Detection of Common Jailbreak Tool URL Schemes

Jailbreak tools often expose custom URL schemes that allow other apps to interact with them. The scanner checks for:

- `cydia://` (Cydia store)
- `sileo://` (Sileo package manager)
- `zbra://` (Zebra package manager)
- `filza://` (Filza file manager)
- Other known jailbreak tool URLs

If these URL schemes are available, the device is likely jailbroken.

4. Identification of Unusual Symbolic Links

Jailbroken devices often create symbolic links (`symlinks`) to system directories that are normally restricted. The scanner inspects:

- `/Applications → /var/stash/Applications/`
- `/Library/Ringtones → /var/stash/Library/Ringtones/`
- `/usr/include → /var/stash/usr/include/`

- `/usr/libexec -> /var/stash/usr/libexec/`

If non-standard symbolic links are detected, the device is likely compromised.

5. Checking Write Access to Restricted System Directories

A non-jailbroken iOS device prevents applications from writing to protected directories. The scanner attempts to:

- Create and delete a test file in:
  - `/private/`
  - `/root/`
  - `/jb/`
- Modify system directories that should be read-only

If write access is successful, the device is jailbroken.

6. Runtime Security Checks

Additional behavioral checks are performed to detect signs of jailbreak:

- **Environment Variables**: Checking for the presence of known jailbreak variables.
- **Process List Scanning**: Searching for suspicious running processes related to jailbreak tools.
- **Debugging Restrictions**: Verifying if the app is running with debugger privileges enabled, which is normally restricted.

JailbreakScanner

Member	Description
static bool IsJailbroken { get; }	Returns `true` if the device is detected as jailbroken, `false` otherwise.
static bool IsEmulator { get; }	Returns `true` if the app running on Emulator, `false` if running on real device.
static bool IsPlatformSupported { get; }	Indicates whether jailbreak detection is available on the current platform (**iOS** only).

Platform Support

JailbreakScanner works on both real iOS devices and the iOS Simulator, but only performs actual detection on real devices. On simulators, it will always return `false` for `IsJailbroken` since simulators cannot be jailbroken in the traditional sense.

Example Usage

```
bool isJailbroken = JailbreakScanner.IsJailbroken;

if (isJailbroken)
{
    // Handle jailbroken device scenario
    // For example, limit functionality or display a warning
    ShowJailbreakWarning();
}
else
{
    // Continue with normal app functionality
    InitializeSecureFeatures();
}
```