Anatomy of Lockdown Mode

0x41con ‘23

whoami

blacktop

- Sr. Security Researcher on ~~Azimuth~~ Trenchant iOS VR team
  - Twitter: @blacktop__
  - Discord: blacktop__ 🤬
  - Mastodon: @blacktop@mastodon.social
  - Bluesky: @blacktop.bsky.social
- AKA That annoying `ipsw` person

Introduction

“Lockdown Mode is an optional, extreme protection that’s designed for the very few individuals who, because of who they are or what they do, might be personally targeted by some of the most sophisticated digital threats. Most people are never targeted by attacks of this nature.”

- Apple (*https://support.apple.com/en-us/HT212650*)

How to Activate Lockdown Mode

iOS 16

In the beginning

How is Lockdown Mode implemented?

Lockdown Mode is implemented as the UserDefaults setting `LDMGlobalEnabled` in the GlobalDomain that frameworks and daemons can read from to limit remotely exposed functionality or enable additional security features.

The DSC Dylibs

DSC Dylibs

Find dylibs that read the Lockdown Mode status in the DSC

```
> i
```

LDM DSC Dylibs

- PrivacySettingsUI
- PreferencesUI
- WebKit
- PhotoLibraryServicesCore
- IMSharedUtilities
- MessagesBlastDoorSupport
- SoftwareUpdateServices
- StatusKitAgentCore
- FaceTime
- TelephonyUtilities
- Email

LDM DSC Dylibs

- PrivacySettingsUI
- PreferencesUI
- WebKit
- PhotoLibraryServicesCore
- IMSharedUtilities
- MessagesBlastDoorSupport
- SoftwareUpdateServices
- StatusKitAgentCore
- FaceTime
- TelephonyUtilities
- Email

PrivacySettingsUI

The framework that handles the Privacy & Security UI section of the Settings.app

The LDMGlobalEnabled setting is checked in the function:

```
bool isLockdownModeEnabled(__int64 a1, const char *a2)
{
  v2 = objc_msgSend_standardUserDefaults_3(&OBJC_CLASS___NSUserDefaults, a2);
  v3 = (void *)objc_claimAutoreleasedReturnValue_71(v2);
  v5 = objc_msgSend_objectForKey_inDomain__1(v3, v4, CFSTR("LDMGlobalEnabled"), CFSTR("NSGlobalDomain"));
  v6 = (void *)objc_claimAutoreleasedReturnValue_71(v5);
  v8 = objc_msgSend_boolValue_4(v6, v7);
  v9 = (char)v8;
  v10 = objc_release_x20_76(v8);
  objc_release_x19_77(v10);
  return v9;
}
```

You can see *(as was mentioned above)* that this is essentially doing:

```
UserDefaults.standard.bool(forKey: "LDMGlobalEnabled")
```

PrivacySettingsUI

The LDMGlobalEnabled setting is also checked in the function:

```
setLockdownModeEnabled(void *a1)  {
  v2 = objc_msgSend_standardUserDefaults(&OBJC_CLASS___NSUserDefaults);
  v3 = (void *)objc_claimAutoreleasedReturnValue_23(v2);
  v4 = objc_msgSend_numberWithBool_(&OBJC_CLASS___NSNumber, UNUSED_ARG(), a1);
  v5 = (void *)objc_claimAutoreleasedReturnValue_23(v4);
  v6 = objc_msgSend_setObject_forKey_inDomain_(v3, UNUSED_ARG(), v5, CFSTR("LDMGlobalEnabled"), CFSTR("NSGlobalDomain"));
  v7 = objc_release_x21_28(v6);
  objc_release_x20_28(v7);
  v9[0] = (__int64)OBJC_CLASS_____NSStackBlock__;
  v9[1] = 3221225472LL;
  v9[2] = (__int64)__setLockdownModeEnabled_block_invoke;
  v9[3] = (__int64)&__block_descriptor_33_e19___NSDictionary_8__0l;
  v10 = (char)a1;
  return AnalyticsSendEventLazy_2(CFSTR("com.apple.security.lockdownmode.status"), v9);
}
```

NOTE: The addition of the com.apple.security.lockdownmode.status Analytics Event being sent on activation.

PrivacySettingsUI

/System/Library/PrivateFrameworks/CoreAnalytics.framework/defaultConfig.json


```
{
  "addTransform": {
    "uuid": "68797474-a5f0-4b03-966f-3f5f9bd640d5_7",
    "name": "lockdownModeStatusAggregateView",
    "inputs": [
      {
        "event": "com.apple.security.lockdownmode.status"
      }
    ],
    "dimensions": [
      {
        "name": "result",
        "type": "boolean",
        "expression": [
          "result",
          ":getEventField"
        ]
      }
    ],
    "measures": [
      {
        "name": "Count",
        "type": "int",
        "function": "count"
      }
    ],
    "outputs": [
      {
        "message": "lockdownModeStatusAggregateView",
        "sink": "Daily",
        "correlation_group_uuid": "b4b16f62-c520-44a8-b1b4-9cbb92
        "correlation_group_name": "SEAR"
      }
    ],
    "type": "aggregate"
  }
}
```

PrivacySettingsUI

The **setter** is called in the ObjC function:

```
-[PUILockdownModeController setLockdownModeGloballyEnabled:]
```

```
> i
```

PrivacySettingsUI

Summary

Enabling Lockdown Mode in the Settings.app Will Do the Following

1. Disable Shared Albums
2. Disable Link Previews *(these are the HTML previews that show up in iMessage when you share a link)*
3. Disable Developer Mode
4. Enable USB Restricted Mode
5. Disable Profile Installation *(this includes installing the Beta Profile to install Beta iOS versions on your device)*
6. Set the LDMGlobalEnabled global user setting to TRUE for other frameworks to check
    - This also sends a com.apple.security.lockdownmode.status analytics event
7. Synchronize the preferences *(alerts all interested parties that the preferences have changed)*
    - Calls CFPreferencesSynchronize("kCFPreferencesAnyApplication", "kCFPreferencesAnyUser", "kCFPreferencesAnyHost");
8. Notify Captive Portal clients *(alerts all WebContents that the lockdown mode has changed and reload the session in Lockdown)*
    - ⚠️ In WebKit captive portal and lockdown mode are essentially the same thing and linked together
9. Reboot the device

Welcome

To your NEW secure iPhone 🎉

LDM DSC Dylibs

- PrivacySettingsUI
- PreferencesUI
- WebKit
- PhotoLibraryServicesCore
- IMSharedUtilities
- MessagesBlastDoorSupport
- SoftwareUpdateServices
- StatusKitAgentCore
- FaceTime
- TelephonyUtilities
- Email

PreferencesUI

The framework that handles the Preferences UI in the Settings.app

The `LDMGlobalEnabled` setting is check has been *inlined* in the ObjC function:

```
-[PSUIPasscodeLockController specifiers]
```

Here is a (cleaned up) snippet:

```
LDMGlobalEnabled = [NSUserDefaults objectForKey:@"LDMGlobalEnabled" inDomain:@"NSGlobalDomain"];
PKIsUSBRestrictedModeDisabledByMobileAsset(args..., LDMGlobalEnabled);
if LDMGlobalEnabled {
  [specifierForID_ALLOW_ACCESS_WHEN_LOCKED setProperty:PSUI_LocalizedStringForPasscodeLock(@"ACCESSORIES_ON")
forKey:@"footerText"];
}
```

Notice in /System/Library/PrivateFrameworks/PreferencesUI.framework/LockdownMode.loctable

```
"en" => {
  "ACCESSORIES_OFF" => {
    "NSStringDeviceSpecificRuleType" => {
      "ipad" => "Accessories are blocked from connecting to your iPad when Lockdown Mode is turned on."
      "iphone" => "Accessories are blocked from connecting to your iPhone when Lockdown Mode is turned on."
      "ipod" => "Accessories are blocked from connecting to your iPod when Lockdown Mode is turned on."
    }
  }
}
```

PreferencesUI

You can see in the screen shot that this is setting the text that appears below the Accessories toggle setting.

PreferencesUI

Further down in the instance method -[PSUIPasscodeLockController specifiers] we also can see:

```
AccessoriesPreferenceSpecifier = [PSSpecifier preferenceSpecifierNamed:@"Accessories",
                                              target:self,
                                              set:@"setEnabledInLockScreenForUSB:specifier:",
                                              get:@"enabledInLockScreenForUSB:",
                                              detail:0,
                                              cell:6,
                                              edit:0];
[AccessoriesPreferenceSpecifier setProperty:@"ACCESSORIES" forKey:@"id"];
[AccessoriesPreferenceSpecifier setProperty:@"allowUSBRestrictedMode" forKey:@“MC_FEATURE"];
USBRestrictedModeLocked = [MCProfileConnection isBoolSettingLockedDownByRestrictions:@“allowUSBRestrictedMode”];
USBRestrictedModeLocked_OR_LockdownModeEnabled = [NSNumber numberWithInt:((USBRestrictedModeLocked |
LDMGlobalEnabled) ^ 1)];
[AccessoriesPreferenceSpecifier setProperty: USBRestrictedModeLocked_OR_LockdownModeEnabled forKey:@"enabled"];
```

This is the code that is disabling the Accessories toggle setting so it cannot be changed while Lockdown Mode is enabled.

LDM DSC Dylibs

- PrivacySettingsUI
- PreferencesUI
- WebKit
- PhotoLibraryServicesCore
- IMSharedUtilities
- MessagesBlastDoorSupport
- SoftwareUpdateServices
- StatusKitAgentCore
- FaceTime
- TelephonyUtilities
- Email

WebKit

https://github.com/WebKit/WebKit

The string LDMGlobalEnabled is assigned in the src file _WKSystemPreferencesInternal.h to the WKSystemPreference LDMEnabledKey

This is only referenced in the file _WKSystemPreferences.mm where we have to following code:

```
+ (BOOL)isCaptivePortalModeEnabled
{
    auto key = adoptCF(CFStringCreateWithCString(kCFAllocatorDefault, WKLockdownModeEnabledKey.characters(), kCFStringEncodingUTF8));
    auto preferenceValue = adoptCF(CFPreferencesCopyValue(key.get(), kCFPreferencesAnyApplication, kCFPreferencesCurrentUser, kCFPreferencesAnyHost));
    if (preferenceValue.get() == kCFBooleanTrue)
        return true;

    key = adoptCF(CFStringCreateWithCString(kCFAllocatorDefault, LDMEnabledKey, kCFStringEncodingUTF8));
    preferenceValue = adoptCF(CFPreferencesCopyValue(key.get(), kCFPreferencesAnyApplication, kCFPreferencesCurrentUser, kCFPreferencesAnyHost));
    return preferenceValue.get() == kCFBooleanTrue;
}
```

This method checks the CFPreferences for the key WKLockdownModeEnabled or LDMGlobalEnabled and returns True if either of them are set to True.

NOTE: It appears Lockdown Mode has been folded into Captive Portal Mode, meaning it can be thought of the same way as Captive Portal Mode in most cases.

WebKit

What does Lockdown Mode/Captive Portal Mode disable?

- WebAssembly
- MP3 Playback
- MathML
- Web Audio API
- Gamepad API
- JPEG 2000
- WebGL
- Speech Recognition API
- RTCDataChannel
- MediaDevices.getUserMedia()
- PDF Viewer
- SVG Fonts
- JIT
- ServiceWorkers

This can be seen in the function adjustSettingsForLockdownMode in the file Source/WebKit/WebProcess/WebPage/WebPage.cpp

```
static void adjustSettingsForLockdownMode(Settings& settings, const WebPreferencesStore& store)
{
    // Disable unstable Experimental settings, even if the user enabled them for local use.
    settings.disableUnstableFeaturesForModernWebKit();
    Settings::disableGlobalUnstableFeaturesForModernWebKit();

    settings.setWebGLEnabled(false);
#if ENABLE(WEBGL2)
    settings.setWebGL2Enabled(false);
#endif
#if ENABLE(GAMEPAD)
    settings.setGamepadsEnabled(false);
#endif
#if ENABLE(WIRELESS_PLAYBACK_TARGET)
    settings.setRemotePlaybackEnabled(false);
#endif
    settings.setFileSystemAccessEnabled(false);
    settings.setAllowsPictureInPictureMediaPlayback(false);
#if ENABLE(PICTURE_IN_PICTURE_API)
    settings.setPictureInPictureAPIEnabled(false);
#endif
    settings.setSpeechRecognitionEnabled(false);
#if ENABLE(SPEECH_SYNTHESIS)
    settings.setSpeechSynthesisAPIEnabled(false);
#endif
#if ENABLE(NOTIFICATIONS)
    settings.setNotificationsEnabled(false);
#endif
#if ENABLE(SERVICE_WORKER)
    settings.setPushAPIEnabled(false);
#endif
#if ENABLE(WEBXR)
    settings.setWebXREnabled(false);
    settings.setWebXRAugmentedRealityModuleEnabled(false);
#endif
#if ENABLE(MODEL_ELEMENT)
    settings.setModelElementEnabled(false);
#endif
#if ENABLE(MEDIA_STREAM)
    settings.setMediaDevicesEnabled(false);
#endif
#if ENABLE(WEB_AUDIO)
    settings.setWebAudioEnabled(false);
#endif
    settings.setDownloadableBinaryFontAllowedTypes(DownloadableBinaryFontAllowedTypes::Restricted);
#if ENABLE(WEB_CODECS)
    settings.setWebCodecsEnabled(false);
    settings.setWebCodecsAV1Enabled(false);
#endif
#if ENABLE(WEB_RTC)
    settings.setPeerConnectionEnabled(false);
    settings.setWebRTCEncodedTransformEnabled(false);
#endif
#if ENABLE(MATHML)
    settings.setMathMLEnabled(false);
#endif
#if ENABLE(PDFJS)
    settings.setPdfJSViewerEnabled(true);
#endif
#if USE(SYSTEM_PREVIEW)
    settings.setSystemPreviewEnabled(false);
#endif
    settings.setEmbedElementEnabled(false);
```

WebKit

JIT?

I knew JIT was disabled, but had a hell of a time determining where in the source code that was being enforced until I back tracked ALL the way up to the top to ProcessLauncher::launchProcess() in ProcessLauncherCocoa.mm

There we can see:

- `launchProcess()`
  - `serviceName(m_launchOptions, m_client);`
    - `webContentServiceName(launchOptions.nonValidInjectedCodeAllowed, client);`

```
static const char* webContentServiceName(bool nonValidInjectedCodeAllowed, ProcessLauncher::Client* client)
{
    if (client && client->shouldEnableLockdownMode())
        return "com.apple.WebKit.WebContent.CaptivePortal";
    return nonValidInjectedCodeAllowed ? "com.apple.WebKit.WebContent.Development" : "com.apple.WebKit.WebContent";
}
```

If Lockdown Mode is enabled then the WebKit launcher will start a com.apple.WebKit.WebContent.CaptivePortal process instead of the normal com.apple.WebKit.WebContent process.

You can also see what entitlements these different services get assigned in the file Source/WebKit/Scripts/process-entitlements.sh

WebKit

What's the difference you ask?

```diff
diff --git a/tmp/WebContent.plist b/tmp/WebContent.CaptivePortal.plist
--- a/tmp/WebContent.plist
+++ b/tmp/WebContent.CaptivePortal.plist
@@ -10,6 +10,15 @@
	<true/>
	<key>com.apple.developer.coremedia.allow-alternate-video-decoder-selection</key>
	<true/>
+	<key>com.apple.developer.kernel.extended-virtual-addressing</key>
+	<true/>
+	<key>com.apple.imageio.allowabletypes</key>
+	<array>
+		<string>org.webmproject.webp</string>
+		<string>public.jpeg</string>
+		<string>public.png</string>
+		<string>com.compuserve.gif</string>
+	</array>
	<key>com.apple.mediaremote.set-playback-state</key>
	<true/>
	<key>com.apple.pac.shared_region_id</key>
@@ -36,8 +45,6 @@
	<array>
		<string>EnableMachBootstrap</string>
	</array>
-	<key>com.apple.private.verified-jit</key>
-	<true/>
	<key>com.apple.private.webinspector.allow-remote-inspection</key>
	<true/>
	<key>com.apple.private.webinspector.proxy-application</key>
@@ -51,8 +58,6 @@
		<string>kTCCServiceCamera</string>
		<string>kTCCServiceMicrophone</string>
	</array>
-	<key>dynamic-codesigning</key>
-	<true/>
</dict>
</plist>
```

So the process itself does NOT even have the entitlements to do JIT if it wanted to.

You can also see they are using entitlements to limit the kinds of images that are supported via ImageIO: webp, jpeg, png, and gif.

Sandbox

As Apple moves to SANDBOX_VERSION_3, they appear to be moving this functionality into the sandbox profile enabling it by using the LockdownModeEnabled sandbox state-flag.

I'm very curious to see what the new EnableExperimentalSandbox and EnableExperimentalSandboxWithProbability state-flag will do 👀

UPDATE: they seemed to have removed them 😞

LDM DSC Dylibs

- PrivacySettingsUI
- PreferencesUI
- WebKit
- PhotoLibraryServicesCore
- IMSharedUtilities
- MessagesBlastDoorSupport
- SoftwareUpdateServices
- StatusKitAgentCore
- FaceTime
- TelephonyUtilities
- Email

PhotoLibraryServicesCore

The `LDMGlobalEnabled` setting is check in the block invocation:

```
__PLIsLockdownMode_block_invoke()
```

Which checks the setting and sets the result to global PLIsLockdownMode_isLockdownMode

That function is called by the stub function PLIsLockdownMode(), which lives in one of the DSC stub island subcaches.

This in turn is called by the stub function PLCanEnableSharedStreamsForAccount() which is implemented in the PhotoLibraryServices framework.

Here is a snippet:

```
return provisionedDataclassesContainsObjectAccountDataclassSharedStreams \
      | ~PLPlatformSharedStreamsSupported()
      | IsRamped
      | MCFeatureSharedStreamAllowed == 2
      | ~MGGetBoolAnswer("PhotoSharingCapability")
      | PLIsEDUMode()
      | PLIsLockdownMode();
```

That in turn is called by the function -[PhotosAccountNotificationPlugin account:willChangeWithType:inStore:oldAccount:] in the PhotosAccountNotificationPlugin bundle.

This corresponds to Apple's summary mentioned above when Shared Albums are removed from the Photos app, and new Shared Album invitations are blocked.

LDM DSC Dylibs

- PrivacySettingsUI
- PreferencesUI
- WebKit
- PhotoLibraryServicesCore
- IMSharedUtilities
- MessagesBlastDoorSupport
- SoftwareUpdateServices
- StatusKitAgentCore
- FaceTime
- TelephonyUtilities
- Email

IMSharedUtilities

iMessage shared utilities framework

The LDMGlobalEnabled setting is read in the ObjC function: +[IMDeviceUtilities deviceIsLockedDown]

Which is called via _objc_msgSend$deviceIsLockedDown from the following functions:

```
> i
```

```
› ipsw dyld macho dyld_shared_cache_arm64e IMSharedUtilities —objc
-V

@interface IMDeviceUtilities : NSObject { // 0x1dd89fc88
/* class methods */

// 0x1a097ee40
+ (BOOL)supportsBusinessChat;
// 0x1a097ee64
+ (BOOL)supportsSpotlight;
// 0x1a097ec64
+ (BOOL)IMDeviceIsGreenTea;
// 0x1a097eeac
+ (BOOL)supportsFaceTime;
// 0x1a097eba8
+ (BOOL)IMDeviceIsChinaRegion;
// 0x1a097ef3c
+ (BOOL)supportsiMessageJunk;
// 0x1a097eed0
+ (BOOL)supportsNameAndPhoto;
// 0x1a097ed98
+ (BOOL)deviceIsLockedDown;
// 0x1a097ee1c
+ (BOOL)supportsDataDetectors;
// 0x1a097ee88
+ (BOOL)supportsSharedWithYou;
// 0x1a097ef18
+ (BOOL)supportsScreenEffects;
// 0x1a097eef4
+ (BOOL)supportsFocus;
// 0x1a097ef50
+ (void)setSupportsiMessageJunk:(BOOL)arg1;
// 0x1a097ed8c
+
(BOOL)IMDeviceRegionIsEligibleToBeForcedIntoFilteringUnknown
Sender;
// 0x1a097eb3c
+ (BOOL)IMSupportsASTC;
// 0x1a097ecd0
+ (BOOL)IMDeviceIsIndiaRegion;
@end
```

NOTE: the supports functions are **true** IF the device is not in Lockdown Mode.

LDM DSC Dylibs

- PrivacySettingsUI
- PreferencesUI
- WebKit
- PhotoLibraryServicesCore
- IMSharedUtilities
- MessagesBlastDoorSupport
- SoftwareUpdateServices
- StatusKitAgentCore
- FaceTime
- TelephonyUtilities
- Email

MessagesBlastDoorSupport

The LDMGlobalEnabled setting is read in the Swift function:

```
(extension in
MessagesBlastDoorSupport):__C.NSUserDefaults.isMessagesLockdownModeEnabled.getter :
Swift.Bool
```

Which is called from the Swift function:

```
MessagesBlastDoorSupport.MessagesBlastDoorInterface.init() ->
MessagesBlastDoorSupport.MessagesBlastDoorInterface
```

MessagesBlastDoorSupport

This function will launch via a dispatch queue either:

- com.apple.MessagesAirlockService if Lockdown Mode is TRUE
- com.apple.MessagesBlastDoorService if Lockdown Mode is FALSE

MessagesAirlockService being the more restrictive, in what kinds of messages/file types can be sent via iMessage, of the two.

MessagesAirlockService is also less than half the size of MessagesBlastDoorService.

LDM DSC Dylibs

- PrivacySettingsUI
- PreferencesUI
- WebKit
- PhotoLibraryServicesCore
- IMSharedUtilities
- MessagesBlastDoorSupport
- SoftwareUpdateServices
- StatusKitAgentCore
- FaceTime
- TelephonyUtilities
- Email

SoftwareUpdateServices

The LDMGlobalEnabled setting is read in the ObjC function:

```
+[SUUtility isLockdownModeEnabled]
```

Which is called via a objc_msgSend stub from the ObjC function:

```
+[SUUtility autoDownloadTimeInterval] {
  if ( [SUUtility isEqualToString:@“Internal”]) && ![SUPreferences.sharedInstance
scanWeeklyInternally:v4] )
  {
    SULogInfo(@“[Auto scan] Apple Internal: Downloading every 2 hours");
    interval = 7200.0;
  }
  else
  {
    if ( [SUUtility isLockdownModeEnabled])
      SULogInfo(@“[Auto scan]: Lockdown mode enabled: Downloading every 1 day");
    else
      SULogInfo(@“[Auto scan] Beta: Downloading every 1 day");
    interval = 86400.0;
  }
  return interval;
}
```

SoftwareUpdateServices

The isLockdownModeEnabled is also called from the following binaries in the Filesystem DMG:

- /usr/libexec/mc_mobile_tunnel
- /System/Library/PrivateFrameworks/SoftwareUpdateServices.framework/Support/softwareupdateservicesd

SoftwareUpdateServices

mc_mobile_tunnel

isLockdownModeEnabled is called in the function:

```
-[MTIPCUTunnelParser _commandInstallProfile:]
```

Here is a snippet from IDA:

```
if ( [MTIPCUTunnelParser _isLockdownModeEnabled] )
{
  v16 = DMCErrorArray(CFSTR("ERROR_PROFILE_INSTALL_PREVENTED_BY_LOCKDOWN"));
  v17 = (id)objc_claimAutoreleasedReturnValue(v16);
  v18 = DMCErrorTypeFatal;
  v19 = DMCInstallationErrorDomain;
  v20 = 4021LL;
}
else
{
  v27 = DMCErrorArray(CFSTR("ERROR_PROFILE_NO_INTERACTIVE_INSTALLATION"));
  v17 = (id)objc_claimAutoreleasedReturnValue(v27);
  v18 = DMCErrorTypeFatal;
  v19 = DMCInstallationErrorDomain;
  v20 = 4020LL;
}
```

This is where it's preventing the installation of profiles when Lockdown Mode is enabled as mentioned by Apple.

SoftwareUpdateServices

softwareupdateservicesd

isLockdownModeEnabled is called in the function:

```
-[SUManagerCore shouldShowRollbackSuggestionAlert:error:]
```

If Lockdown Mode is enabled, it will log "Device is in Lockdown Mode. Ignoring revoked update" and NOT allow a rollback from a revoted update.

The above function is called via an objc_msgSend stub by the following functions:

- `[SUManagerCore revokedUpdateFound:]`
  - Where LDM determines if Rollback Suggestions are shown.
- `[SUManagerCore suggestRollback:rollbackDescriptor:withResult:]`
  - Where LDM prevents the presentation of the rollback.

It is possible this will prevent rolling back the NEW Rapid Security Response (RSR)

LDM DSC Dylibs

- PrivacySettingsUI
- PreferencesUI
- WebKit
- PhotoLibraryServicesCore
- IMSharedUtilities
- MessagesBlastDoorSupport
- SoftwareUpdateServices
- StatusKitAgentCore
- FaceTime
- TelephonyUtilities
- Email

StatusKitAgentCore

The LDMGlobalEnabled setting is read in the ObjC function:

```
-[SKAStatusServer _inTextTrafficMode]
```

Which is called via a objc_msgSend stub from the ObjC function:

```
-[SKAStatusServer init]
```

Here is a snippet:

```
inTextTrafficMode = (unsigned int)objc_msgSend__inTextTrafficMode(SKAStatusServer, v18);
SKAStatusServer->_trafficModeEnabled = inTextTrafficMode;
if ( inTextTrafficMode )
{
  v21 = objc_msgSend_logger_0(&OBJC_CLASS___SKAStatusServer, v20);
  v22 = (NSObject *)objc_claimAutoreleasedReturnValue_645(v21);
  if ( os_log_type_enabled_679(v22, OS_LOG_TYPE_DEFAULT) )
  {
    v79[0] = 0;
    _os_log_impl_605(
      &dword_1F7002000,
      v22,
      OS_LOG_TYPE_DEFAULT,
      "In Traffic Mode--will not honor status subscriptions and will drop incoming status's",
      (uint8_t *)v79,
      2u);
  }
  ((void (*)(void))objc_release_x20_734)();
}
```

StatusKitAgentCore

To find other uses of the _trafficModeEnabled *ivar*:

```
› ipsw dyld search dyld_shared_cache_arm64e --ivar _trafficModeEnabled

StatusKitAgentCore  class=SKAInvitationManager                ivar=_trafficModeEnabled
StatusKitAgentCore  class=SKAStatusSubscriptionServiceClient ivar=_trafficModeEnabled
StatusKitAgentCore  class=SKAPushManager                      ivar=_trafficModeEnabled
StatusKitAgentCore  class=SKAStatusServer                     ivar=_trafficModeEnabled
```

It appears to use the Lockdown Mode status to determine if it should enable the trafficModeEnabled property or the SKAStatusServer and when enabled will drop all incoming status's even if you've subscribed to them.

StatusKitAgentCore

StatusKitAgentCore appears to not be imported by any other dylibs in the dyld_shared_cache_arm64e:

```
❯ ipsw dyld imports dyld_shared_cache_arm64e StatusKitAgentCore

StatusKitAgentCore Imported By:
===============================

In DSC (Dylibs)
---------------
  • None

In FileSystem DMG (Apps)
------------------------
  • /System/Library/PrivateFrameworks/StatusKit.framework/StatusKitAgent
  • /cdhash/d4503a059e8f571354e50e8af8713ba264a52874 (/System/Library/PrivateFrameworks/StatusKit.framework/StatusKitAgent)
```

According to the manpage for StatusKitAgent it is the agent backing StatusKit.framework and was introduced in macOS 12 and appears to be Apple's local and iCloud event pubsub private framework.

com.apple.private.alloy.status.keysharing StatusKit Keysharing "Your focus won't be shared with other people in Lockdown Mode."

LDM DSC Dylibs

- PrivacySettingsUI
- PreferencesUI
- WebKit
- PhotoLibraryServicesCore
- IMSharedUtilities
- MessagesBlastDoorSupport
- SoftwareUpdateServices
- StatusKitAgentCore
- FaceTime
- TelephonyUtilities
- Email

FaceTime

The LDMGlobalEnabled setting is read in the ObjC function:

```
-[FaceTimePushHandler
handler:didReceiveMessage:forTopic:fromID:messageContext:]
```

```
if ( LDMGlobalEnabled )
{
  if ( os_log_type_enabled_68(v13, OS_LOG_TYPE_DEFAULT) )
  {
    _os_log_impl_61(
      &dword_20A04C000,
      (NSObject *)OSLogHandleForIDSCategory("FTMessageDelivery"),
      OS_LOG_TYPE_DEFAULT,
      "dropping push message: %@ for IMAV because lockdown mode is enabled",
      (uint8_t *)&138412290,
      0xC);
  }
  if ( (unsigned int)os_log_shim_legacy_logging_enabled() )
  {
    if ( MarcoShouldLog(CFSTR("Service")) )
      MarcoLog(
        0,
        CFSTR("Service"),
        CFSTR("IMAV Lockdown Mode"),
        CFSTR("dropping push message: %@ for IMAV because lockdown mode is enabled"));
    if ( IMShouldLog_0(CFSTR("IDS"), 0) )
      IMLogString(
        0,
        CFSTR("IDS"),
        CFSTR("Service"),
        CFSTR("FTMessageDelivery"),
        CFSTR("dropping push message: %@ for IMAV because lockdown mode is enabled"));
  }
}
```

I believe this is FaceTime dropping incoming FaceTime calls unless you have previously called that person or contact.

LDM DSC Dylibs

- PrivacySettingsUI
- PreferencesUI
- WebKit
- PhotoLibraryServicesCore
- IMSharedUtilities
- MessagesBlastDoorSupport
- SoftwareUpdateServices
- StatusKitAgentCore
- FaceTime
- TelephonyUtilities
- Email

TelephonyUtilities

🆕 in iOS 16.1

The LDMGlobalEnabled setting is read in the function:

```
TULockdownModeEnabled();
```

That function is queried in the following 2 functions:

- `-[TUCall initWithUniqueProxyIdentifier:endpointOnCurrentDevice:notificationCenter:]`
  - appears to just set the TUCall->lockdownModeEnabled property.
- block func that invokes `-[TUMomentsController initWithDelegate:dataSource:]`
  - which just sets the TUMomentsController->lockdownModeEnabled property.

This means that anything using TUCall or TUMomentsController will be now be Lockdown Mode aware.

Both classes have getters for the property:

- `-[TUCall lockdownModeEnabled]`
- `-[TUMomentsController lockdownModeEnabled]`

TelephonyUtilities

To see what imports TelephonyUtilities we can run:

```
› ipsw dyld imports dyld_shared_cache_arm64e TelephonyUtilities
```

⚠️ TelephonyUtilities is imported in 77 DSC dylibs, and 73 binaries in the Filesystem DMG

I was not able to find ANY uses of the getters in the DSC dylibs, but I did find a few uses in the Filesystem DMG:

```
› ipsw macho search -i iPhone15,2_16.3_20D47_Restore.ipsw --sym _TULockdownModeEnabled

  /System/Library/PrivateFrameworks/TelephonyUtilities.framework/XPCServices/com.apple.FTLivePhotoService.xpc/com.apple.FTLivePhotoService
  /System/Library/PrivateFrameworks/TelephonyUtilities.framework/callservicesd
```

TelephonyUtilities

com.apple.FTLivePhotoService

com.apple.FTLivePhotoService imports TULockdownModeEnabled

Where in -[ServiceDelegate listener:shouldAcceptNewConnection:] LDM can lead to the following failure:

"Rejecting the connection since it is not entitled %@, lockdownModeEnabled: %@"

This seem to be related to FaceTime and it is my guess that the lockdownModeEnabled property is used to prevent FaceTime calls from strangers and prevent Live Photos from being sent.

TelephonyUtilities

callservicesd

callservicesd has 2 ObjC classes that use the lockdownModeEnabled selector

```
› ipsw macho search -i iPhone15,2_16.3_20D47_Restore.ipsw --sel lockdownModeEnabled
  0x1000182f8: /System/Library/PrivateFrameworks/TelephonyUtilities.framework/callservicesd (CSDUserActivityManager)
  0x100107094: /System/Library/PrivateFrameworks/TelephonyUtilities.framework/callservicesd (CSDConversationManager)
```

It is used via the objc stub _objc_msgSend$lockdownModeEnabled in the following functions:

- -[CSDUserActivityManager updateHandoffAdvertisementForConversation:fromOldConversation:uplinkMuted:sendingVideo:]
  - "Not updating handoff advertisement because lockdown mode is enabled"
- -[CSDUserActivityManager updateHandoffRegistrationForConversation:fromOldConversation:]
  - "Not updating handoff registration because lockdown mode is enabled"
- -[CSDProviderCall joinConversationWithRequest:]
- -[CSDCall callRelaySupport]
  - Where if lockdownModeEnabled is true, it will return 0
- -[CSDConversationManager updateConversationHandoffEligibility:]
  - "Not updating handoff eligibility %@ because lockdown mode is enabled”
- -[CSDConversation session:endedWithError:reason:]
  - "Unknown participant added whle in lockdown mode, cleaning up"
- -[CSDConversationManager conversation:addedActiveParticipant:]

These all seem to be related to the iOS Handoff/Call Relay features, which allows you to continue a call on another device.

LDM DSC Dylibs

- PrivacySettingsUI
- PreferencesUI
- WebKit
- PhotoLibraryServicesCore
- IMSharedUtilities
- MessagesBlastDoorSupport
- SoftwareUpdateServices
- StatusKitAgentCore
- FaceTime
- TelephonyUtilities
- Email

Email

🆕 in iOS 16.2

The LDMGlobalEnabled setting is read in the ObjC function:

```
+[NSUserDefaults(Email) em_lockdownModeEnabled]
```

```
> i
```

Which in turn is called by 2 other dylibs that import Email:

- Message
- MailWebProcessSupport

Email

Message

em_lockdownModeEnabled is called in the ObjC function:

```
+[MFAttachment isSinglePagePDF:]
```

Where if the device is in Lockdown Mode it disables PDF parsing and logs the following message:

"Lockdown Mode detected, disabling PDF parsing"

Email

MailWebProcessSupport

em_lockdownModeEnabled is called in the ObjC function:

```
-[MSPDFAttachmentController updateToInlinePDFAttachmentIfNeeded:]
```

Where if the device is in Lockdown Mode it disables PDF parsing and logs the following message:

"(%{public}@) Lockdown Mode detected, disabling PDF parsing”

The Filesystem Binaries

LDM Filesystem binaries

Find binaries that read the Lockdown Mode status in the Filesystem DMG

```
❯ ipsw mount fs iPhone15,2_16.6_20G5047d_Restore.ipsw

  • Found Filesystem DMG
    • Created 098-24861-056.dmg
    • Mounting 098-24861-056.dmg
      • Press Ctrl+C to unmount /tmp/098-24861-056.dmg.mount ...

❯ rg -Fla LDMGlobalEnabled /tmp/098-24861-056.dmg.mount

• /System/Library/PrivateFrameworks/IDS.framework/identityservicesd.app/identityservicesd
• /System/Library/PreferenceBundles/AccountSettings/AppleAccountSettings.bundle/AppleAccountSettings
• /System/Library/DataClassMigrators/PreferencesMigrator.migrator/PreferencesMigrator
• /System/Library/CoreServices/ReportCrash
```

LDM Filesystem binaries

- identityservicesd
- AppleAccountSettings
- PreferencesMigrator
- ReportCrash

LDM Filesystem binaries

- identityservicesd
- AppleAccountSettings
- PreferencesMigrator
- ReportCrash

identityservicesd

The LDMGlobalEnabled setting is checked in the ObjC function:

```
+[IMUserDefaults isFirewallEnabled]
```

Where if first checks the NSUserDefaults for com.apple.ids and the key FirewallOn and if that is FALSE, it checks if Lockdown Mode is enabled.

This is called from the following functions:

1. `-[IDSDXPCInternalTesting triggerAllFirewallNotificationsWithHandle:delay:service:]`

   In which if the Firewall is enabled it will send the firewall notifications to the IDS service.

2. `-[IDSDeliveryController _constructDeliveryPipeline]`

   Where if the Firewall is enabled, it will populate the pipeline from the IDSFirewallStore

3. `-[IDSDaemon isFirewallOn]`

   Which is just a thin wrapper and is called by:

   ```
   -[IDSDaemon isFirewallOnForService:]
   ```

   Which enables the Firewall for a given service and is called by `-[IDSDaemon shouldFirewallDropMessageForFromID:mergeID:service:]` and another function that builds and sends IDSLogs.

identityservicesd

4. `-[IDSDaemon isFirewallEnabledForService:]`

   Which is ALSO called by -[IDSDaemon isFirewallOnForService:] *(see number 3)*

5. `sub_100680D9C(id *a1)`

Which mostly seems to be for setting up and logging Firewall settings and contains the strings:

- "Is firewall enabled %@"
- "Should firewall drop for all categories %@"
- "Entries for category %u: %@"

And has the call stack:

- `-[IDSFirewallStore _registerSysdiagnoseBlock]`
  - `-[IDSFirewallStore initWithFilePath:]`
    - `-[IDSFirewallStore init]`

identityservicesd

IDSFirewallLocalizable.strings

```
› plutil -p /System/Library/PrivateFrameworks/IDS.framework/
identityservicesd.app/en.lproj/IDSFirewallLocalizable.strings
```

```
"%@ attempted to set up a share with you." => "%@ attempted to set up a share with you."
"%@ attempted to share a FaceTime Live Photo." => "%@ attempted to share a FaceTime Live Photo."
"%@ attempted to share Apple Card or transaction information." => "%@ attempted to share Apple Card or transaction information."
"%@ attempted to share something using AirDrop." => "%@ attempted to share something using AirDrop."
"%@ attempted to share their ETA with you." => "%@ attempted to share their ETA with you."
"%@ attempted to SharePlay Music." => "%@ attempted to SharePlay Music."
"%@ invited you to a Shared Photo Stream." => "%@ invited you to a Shared Photo Stream."
"%@ invited you to play a game." => "%@ invited you to play a game."
"%@ invited you to share your Car Key." => "%@ invited you to share your Car Key."
"%@ invited you to share your Fitness Activity." => "%@ invited you to share your Fitness Activity."
"%@ invited you to share your health data." => "%@ invited you to share your health data."
"%@ invited you to share your location." => "%@ invited you to share your location."
"%@ invited you to Walkie-Talkie." => "%@ invited you to Walkie-Talkie."
"A teacher attempted to send a command using Classroom." => "A teacher attempted to send a command using Classroom."
"An AppleCare advisor attempted to gather information about your device." => "An AppleCare advisor attempted to gather information about your dev
"An AppleCare advisor attempted to gather information about your iPad." => "An AppleCare advisor attempted to gather information about your iPad
"An AppleCare advisor attempted to gather information about your iPhone." => "An AppleCare advisor attempted to gather information about your iPh
"An AppleCare advisor attempted to gather information about your Mac." => "An AppleCare advisor attempted to gather information about your Mac.
"An AppleCare advisor attempted to view your screen." => "An AppleCare advisor attempted to view your screen."
"An unknown contact attempted to access a camera." => "An unknown contact attempted to access a camera."
"An unknown contact attempted to access a Home." => "An unknown contact attempted to access a Home."
"An unknown contact attempted to collaborate using iCloud." => "An unknown contact attempted to collaborate using iCloud."
"An unknown contact attempted to FaceTime Audio call." => "An unknown contact attempted to FaceTime Audio call."
"An unknown contact attempted to FaceTime." => "An unknown contact attempted to FaceTime."
"An unknown contact attempted to Group FaceTime." => "An unknown contact attempted to Group FaceTime."
"An unknown contact attempted to invite you to a Home." => "An unknown contact attempted to invite you to a Home."
"An unknown contact attempted to request or send a payment." => "An unknown contact attempted to request or send a payment."
"An unknown contact attempted to request purchase authorization." => "An unknown contact attempted to request purchase authorization."
"An unknown contact attempted to screen share with you." => "An unknown contact attempted to screen share with you."
"An unknown contact attempted to set up a share with you." => "An unknown contact attempted to set up a share with you."
"An unknown contact attempted to share a FaceTime Live Photo." => "An unknown contact attempted to share a FaceTime Live Photo."
"An unknown contact attempted to share Apple Card or transaction information." => "An unknown contact attempted to share Apple Card or transact
"An unknown contact attempted to share something using AirDrop." => "An unknown contact attempted to share something using AirDrop."
"An unknown contact attempted to share their ETA with you." => "An unknown contact attempted to share their ETA with you."
"An unknown contact attempted to SharePlay Music." => "An unknown contact attempted to SharePlay Music."
"An unknown contact invited you to a Shared Photo Stream." => "An unknown contact invited you to a Shared Photo Stream."
"An unknown contact invited you to play a game." => "An unknown contact invited you to play a game."
"An unknown contact invited you to share your Car Key." => "An unknown contact invited you to share your Car Key."
"An unknown contact invited you to share your Fitness Activity." => "An unknown contact invited you to share your Fitness Activity."
"An unknown contact invited you to share your health data." => "An unknown contact invited you to share your health data."
"An unknown contact invited you to share your location." => "An unknown contact invited you to share your location."
"An unknown contact invited you to Walkie-Talkie." => "An unknown contact invited you to Walkie-Talkie."
"Lockdown Mode blocked a Device Inquiry" => "Lockdown Mode blocked a Device Inquiry"
"Lockdown Mode blocked Activity Sharing invitation" => "Lockdown Mode blocked Activity Sharing invitation"
"Lockdown Mode blocked AirDrop" => "Lockdown Mode blocked AirDrop"
"Lockdown Mode blocked Apple Card sharing" => "Lockdown Mode blocked Apple Card sharing"
"Lockdown Mode blocked Apple Pay request" => "Lockdown Mode blocked Apple Pay request"
"Lockdown Mode blocked camera access" => "Lockdown Mode blocked camera access"
"Lockdown Mode blocked Car Key Sharing invitation" => "Lockdown Mode blocked Car Key Sharing invitation"
"Lockdown Mode blocked Classroom" => "Lockdown Mode blocked Classroom"
"Lockdown Mode blocked ETA Sharing" => "Lockdown Mode blocked ETA Sharing"
"Lockdown Mode blocked FaceTime" => "Lockdown Mode blocked FaceTime"
"Lockdown Mode blocked FaceTime Audio" => "Lockdown Mode blocked FaceTime Audio"
"Lockdown Mode blocked FaceTime Live Photo" => "Lockdown Mode blocked FaceTime Live Photo"
"Lockdown Mode blocked Find My" => "Lockdown Mode blocked Find My"
"Lockdown Mode blocked Focus Sharing" => "Lockdown Mode blocked Focus Sharing"
"Lockdown Mode blocked Game Center invitation" => "Lockdown Mode blocked Game Center invitation"
"Lockdown Mode blocked Group FaceTime" => "Lockdown Mode blocked Group FaceTime"
"Lockdown Mode blocked Health Sharing invitation" => "Lockdown Mode blocked Health Sharing invitation"
"Lockdown Mode blocked Home access" => "Lockdown Mode blocked Home access"
"Lockdown Mode blocked Home invitation" => "Lockdown Mode blocked Home invitation"
"Lockdown Mode blocked iCloud collaboration" => "Lockdown Mode blocked iCloud collaboration"
"Lockdown Mode blocked Intercom message" => "Lockdown Mode blocked Intercom message"
```

LDM Filesystem binaries

- identityservicesd
- AppleAccountSettings
- PreferencesMigrator
- ReportCrash

AppleAccountSettings

The LDMGlobalEnabled setting is checked in the ObjC function:

```
-[AAUIDataclassDetailSpecifierProvider isDataclassLDMDisabled]
```

Which is called by:

```
-[AAUIDataclassDetailSpecifierProvider _switchSpecifierWithTitle:]
  preferenceSpecifier = [PSSpecifier preferenceSpecifierNamed:id target:self set:"_setValue:forSpecifier:" get:"_toggleValue
  v5 = [NSNumber numberWithBool:[AAUIDataclassDetailSpecifierProvider isDataclassLDMDisabled] ^ 1];
  [preferenceSpecifier setObject:v5 forKeyedSubscript:PSEnabledKey];
  [preferenceSpecifier setObject:self->_dataclass forKeyedSubscript:PSIDKey];
  [preferenceSpecifier setObject:self->_dataclass forKeyedSubscript:@"com.apple.accountsui.dataclass"];
  return preferenceSpecifier;
```

Which in turn is called by:

```
-[AAUIDataclassDetailSpecifierProvider _headerGroupSpecifiers]
```

```
if ( ![AAUIDataclassDetailSpecifierProvider->_dataclass isEqualToString:"ACAccountDataclassHealth"])
    [AAUIDataclassDetailSpecifierProvider _switchSpecifierWithTitle:[NSBundle bundleForClass:AAUIDataclassDetailSpecifier
  localizedStringForKey:"Sync this iPhone" value:0 table:@“Localizable”];
```

LDM Filesystem binaries

- identityservicesd
- AppleAccountSettings
- PreferencesMigrator
- ReportCrash

PreferencesMigrator

The LDMGlobalEnabled setting is checked in the ObjC function:

```
-[PreferencesDataMigrator _performLockdownModeMigration]
```

"Running Lockdown Mode migration: lockdownModeEnabled=%d"

Which in turn is called by:

```
-[PreferencesDataMigrator performMigration]
```

Here is a snippet:

```
objc_msgSend(self, "_performTigrisSoundSettingsMigration");
objc_msgSend(self, "_performYukonRevertSoundSettingsMigration");
objc_msgSend(self, "_performLocationHomeKitSettingsMigration");
objc_msgSend(self, "_performLocationBasedAlertSettingsMigration");
objc_msgSend(self, "_performLocationRoutingAndTrafficSettingsMigration");
objc_msgSend(self, "_performLocationWifiBluetoothUWBSettingsMigration");
objc_msgSend(self, "_performSystemCustomizationSettingsMigration");
objc_msgSend(self, "_performWallpaperPhotoLibraryMigration");
objc_msgSend(self, "_performImproveMapsSettingsMigration");
objc_msgSend(self, "_performSydneyHTTP3DeveloperOptInMigration");
objc_msgSend(self, "_performLockdownModeMigration"); <----------------
objc_msgSend(self, "_performIsAppAndAccessoryAnalyticsAllowedRemovalMigration");
```

I believe this is use when you are setting up a NEW device and restoring from a backup of a previous device that had Lockdown Mode enabled.

LDM Filesystem binaries

- identityservicesd
- AppleAccountSettings
- PreferencesMigrator
- ReportCrash

ReportCrash

The LDMGlobalEnabled setting is checked in the ObjC function via a block:

```
+[OSACrashReport isInLDM]
```

Which is called in the following 2 functions via an objc_msgSend stub:

```
-[OSACrashReport checkIfNeedsUrgentSubmission]
```

Snippet:

```
if ( (unsigned int)objc_msgSend_isInLDM(v138, v139) )
  (*(void (__fastcall **)(__int64, void **))(block + 16))(block, &off_10002FDB8);
```

This appears to be populating a NSConstantDictionary's ldm key with a value of __kCFBooleanTrue. This will add "ldm": 1 to the crashlogs.

```
-[OSACrashReport checkIfNeedsUrgentSubmission]
```

Snippet:

```
if ( OSACrashReport->_sroute_id ) {
  OSACrashReport->_needsUrgentSubmission = 1;
} else if ( objc_msgSend_isInLDM() ) {
  if ( [[OSACrashReport problemType] isEqualToString:@"309"] ) { // 309 means not lightweight corpse and exceptionType != EXC_RESOURCE
    if ( has_run != -1 ) {
      dispatch_once(&has_run, &create_urgentSubmissionCount_dispatch_queue); // "com.apple.osanalytics.urgentSubmissionCount"
    }
    block[0] = (__int64)_NSConcreteStackBlock;
    block[1] = 0xC0000000LL;
    block[2] = (__int64)setUrgentSubmissionBasedOnDailyCount;
    block[3] = (__int64)&unk_100028880;
    block[4] = (__int64)self;
    dispatch_sync((dispatch_queue_t)urgentSubmissionCount_dispatch_queue, block);
  }
}
```

If Lockdown Mode is enabled *(or if running a beta OS)* this will set OSACrashReport->_needsUrgentSubmission = 1 for the first 75 *(plus a random amount between 0-351)* crashes per App per day.

I assume this marks them for additional scrutiny during analysis.

How to detect LDM?

How to detect LDM?

In the browser

https://github.com/botherder/is-lockdown *(updated to work on macOS as well)*

```
(function(window) {
    function isLockdown() {
        var _lib = {};

        _lib.isWebGLAvailable = function() {
            return !!(window.WebGLRenderingContext);
        }

        _lib.isWebRTCAvailable = function() {
            let rtcFunctions = [
                'RTCIceGatherer',
                'RTCPeerConnection',
                'webkitRTCPeerConnection',
                'mozRTCPeerConnection',
            ];

            var rtcAvailable = false;
            rtcFunctions.forEach(function(item) {
                if (rtcAvailable)
                    return;

                if (item in window)
                    rtcAvailable = true;
            });

            return rtcAvailable;
        }

        _lib.isMathMLAvailable = function() {
            let testId = 'isLockdownMathMLTest';
            let div = document.createElement('div');
            div.setAttribute('id', testId);
            div.innerHTML = '<math style="display: none"><mrow mathcolor=\"red\"><mn>1</mn></mrow></math>';
            document.body.appendChild(div);
            let mathMLAvailable = window.getComputedStyle(div.firstChild.firstChild, null).color === 'rgb(255, 0, 0)';
            document.body.removeChild(document.getElementById(testId));
            return mathMLAvailable;
        }

        _lib.isLockdownAvailable = function() {
            if (/iP(hone|od|ad)/.test(navigator.platform) == true) {
                let version = (navigator.appVersion).match(/OS (\d+)_(\d+)_?(\d+)?/);
                let majorVersion = parseInt(version[1], 10)
                return majorVersion >= 16
            } else if (/MacIntel/.test(navigator.platform) == true) {
                let version = navigator.appVersion.match(/Version\/(\d+)\.(\d+)?\.?(\d+) Safari/);
                let majorVersion = parseInt(version[1], 10)
                return majorVersion >= 16
            } else {
                return false;
            }
        }

        _lib.isLockdownEnabled = function() {
            if (_lib.isLockdownAvailable() && !_lib.isWebGLAvailable()
                && !_lib.isWebRTCAvailable() && !_lib.isMathMLAvailable())
                return true;

            return false;
        }

        return _lib;
    }


    if (typeof(window.isLockdown) === 'undefined') {
        window.isLockdown = isLockdown();
    }
})(window);
```

How to disable LDM

How to disable LDM

Programmatically

In Swift

```swift
import Foundation

// check current status
var isLockDownEnabled = UserDefaults.standard.bool(forKey: "LDMGlobalEnabled")
print(isLockDownEnabled)

// disable Globally
UserDefaults.standard.setPersistentDomain(["LDMGlobalEnabled":false], forName: UserDefaults.globalDomain)
UserDefaults.standard.synchronize()
```

How to disable LDM

macOS

To read the Lockdown Mode status on macOS, you can use the defaults command *(where 1 is active and 0 is inactive)*:

```
defaults read .GlobalPreferences.plist LDMGlobalEnabled
```

To set Lockdown Mode on macOS, you can use the defaults command:

```
# turn lockdown mode off
defaults write .GlobalPreferences.plist LDMGlobalEnabled -bool false
# turn lockdown mode on
defaults write .GlobalPreferences.plist LDMGlobalEnabled -bool true
```

"That’s right, it’s not written to a rootless/SIP protected file like TCC.db" - brunerd

⚠️ When you disable Lockdown Mode in either of the two above mentioned ways it isn't immediately reflected in the Settings.app or Safari. I’ve noticed it takes a little while to take effect

How to disable LDM

In WebKit

To disable LDM you would need to call `[_WKSystemPreferences setCaptivePortalModeEnabled:false]`

This is only called by:

`-[WKWebpagePreferences _setCaptivePortalModeEnabled:enabled]` in WKWebpagePreferences.mm

However, on a IOS_FAMILY device the entitlement com.apple.developer.web-browser is required.

We can see what files contain this entitlement with the following:

```
> i
```

How to disable LDM

In Safari

You are able to disable Lockdown Mode per site via the Safari.app

How to disable LDM

In Safari

Where is this per site data stored?

Grepping around the symbols for lockdownMode and domain I found:

```
> i
```

`-[WBSLockdownModePreferenceManager setLockdownModeEnabled:forDomain:]` sounds like what I need 🤔

How to disable LDM

In Safari

class-dumping the dylib in the DSC with `ipsw`

```
❯ ipsw dyld macho dyld_shared_cache_arm64e SafariShared --objc -V

@interface WBSLockdownModePreferenceManager : WBSPerSitePreferenceManager<WBSPerSitePreferenceManagerDefaultsDelegate, WBSPerSitePreferenceManagerStorageDelegate, WBSPerSitePreferenceBinaryToggleItemManager> { // 0x1de0dd828
  /* instance variables */                                  // +size   offset
  WBSPerSitePreferencesSQLiteStore *_perSitePreferenceStore; // +0x8    0x20
  WBSPerSitePreference *_lockdownPreference;                 // +0x8    0x28
}

@property (readonly, nonatomic) WBSPerSitePreference *lockdownPreference;
@property (readonly, nonatomic) WBSPerSitePreferencesSQLiteStore *perSitePreferencesStore;
@property (readonly) unsigned long long hash;
@property (readonly) Class superclass;
@property (readonly, copy) NSString *description;
@property (readonly, copy) NSString *debugDescription;

/* instance methods */

// 0x1a3f0fbcc
- (id)preferences;
// 0x1a3f0fca4
- (id)onValueForPreference:(id)arg1;
// 0x1a3f0fc7c
- (id)preferenceNameForPreference:(id)arg1;
// 0x1a3f0fc8c
- (id)defaultPreferenceValueForPreferenceIfNotCustomized:(id)arg1;
// 0x1a3f0fcc0
- (void).cxx_destruct;
// 0x1a3f0fc58
- (id)localizedStringForValue:(id)arg1 inPreference:(id)arg2;
// 0x1a3f0fc98
- (id)offValueForPreference:(id)arg1;
// 0x1a3f0fc6c
- (id)perSitePreferencesStore;
// 0x1a3f0fb00
- (void)didUpdatePreference:(id)arg1 toValue:(id)arg2 forDomain:(id)arg3;
// 0x1a3f0fc4c
- (id)valuesForPreference:(id)arg1;
// 0x1a3f0f978
- (void)getLockdownModeEnabledForDomain:(id)arg1 withTimeout:(double)arg2 fallbackEnabledState:(BOOL)arg3 usingBlock:(id /* block */)arg4;
// 0x1a3f0f81c
- (id)initWithPerSitePreferenceStore:(id)arg1;
// 0x1a3f0fcb0
- (id)lockdownPreference;
// 0x1a3f0fc64
- (BOOL)preferenceDoesNotHaveForOtherWebsitesDefault:(id)arg1;
// 0x1a3f0f8e8
- (void)setLockdownModeEnabled:(BOOL)arg1 forDomain:(id)arg2;
@end
```

👉👀

So the WBSLockdownModePreferenceManager seems to be backed by a SQLite database

How to disable LDM

In Safari

With a bit more reversing I found the database locations:

macOS

$HOME/Library/Safari/PerSitePreferences.db

iOS *(com.apple.mobilesafari)*

$CONTAINER/Library/Safari/PerSitePreferences.db

Example:

/private/var/mobile/Containers/Data/Application/D5671234-8C34-4934-BA34-46C9424CP30/Library/Safari/PerSitePreferences.db

Querying the Database

```
› sqlite3 PerSitePreferences.db 'SELECT * FROM "preference_values" WHERE “preference"="PerSitePreferencesLockdownMode";'
```

Setting the Database

```
› sqlite3 PerSitePreferences.db 'INSERT INTO "preference_values" VALUES (NULL,"trenchant.io","PerSitePreferencesLockdownMode",0, NULL);'
```

iOS 17

The future

Here We Go Again

You know the drill by know

Let’s look for LDMGlobalEnabled

```
>
```

Where did they all go?

What about the FS bins?

```
› rg -Fla LDMGlobalEnabled /tmp/096-01403-035.dmg.mount/

  /System/Library/PrivateFrameworks/IDS.framework/identityservicesd.app/identityservicesd
```

What the hell is going ON here?!

Quick Look at LockdownMode Framework

First instead of LDMGlobalEnabled being a CFString it is now a

```
static LockdownMode.Constants.userDefaultsKey.getter : Swift.String
```

This is of course because it is written in Swift… 🤮

The userDefaultsKey.getter is checked in the function:

```
LockdownMode.LockdownModeManager.enabled.getter : Swift.Bool
```

Interestingly, however, the function now checks the sysctl security.mac.lockdown_mode_state

If that returns 0, then it falls back to the old way of checking the

```
UserDefaults.standard.bool(forKey: "LDMGlobalEnabled")
```

You will see the LockdownModeManager.shared.enabled used typically by things that import the framework.

Missing FWs

I've noticed that not all frameworks have converted over to using the NEW LockdownMode framework.

Those that do not instead use it via dlopen

```
› ipsw dyld str -p '/System/Library/PrivateFrameworks/LockdownMode.framework/LockdownMode' dyld_shared_cache_arm64e


• Searching for strings...
0x19d4956a0: /System/Library/PrivateFrameworks/LockdownMode.framework/LockdownMode                WebKit
0x19f806ecf: softlink:r:path:/System/Library/PrivateFrameworks/LockdownMode.framework/LockdownMode PhotoLibraryServicesCore
0x19fec2f01: softlink:r:path:/System/Library/PrivateFrameworks/LockdownMode.framework/LockdownMode TelephonyUtilities
0x1ebf54cff: softlink:r:path:/System/Library/PrivateFrameworks/LockdownMode.framework/LockdownMode PrivacySettingsUI
0x218e2e2b0: softlink:r:path:/System/Library/PrivateFrameworks/LockdownMode.framework/LockdownMode PreferencesUI
```

Foreshadowing

Some NEW strings in /System/Library/PrivateFrameworks/Settings/PrivacySettingsUI.framework/LockdownMode.loctable

```
"SECTION_WIRELESS_CONNECTIVITY_HEADER" => "Wireless Connectivity"
"SECTION_WIRELESS_CONNECTIVITY_CONTENT" => "Automatically joining non-secure Wi-Fi networks is blocked."

"ENABLE_2G" => "2G Cellular Connectivity"
"ENABLE_2G_FOOTER" => "For enhanced security, 2G cellular connectivity is turned off when in Lockdown Mode. Emergency calls may temporarily turn on 2G, if needed."
```

What imports the LockdownMode framework?

```
> i
```

NEW dylibs

- /System/Library/Frameworks/LinkPresentation.framework/LinkPresentation
- /System/Library/PrivateFrameworks/WiFiKitUI.framework/WiFiKitUI
- /System/Library/PrivateFrameworks/WiFiKit.framework/WiFiKit

NEW dylibs

- /System/Library/Frameworks/LinkPresentation.framework/LinkPresentation
- /System/Library/PrivateFrameworks/WiFiKitUI.framework/WiFiKitUI
- /System/Library/PrivateFrameworks/WiFiKit.framework/WiFiKit

LinkPresentation

LDM is checked in the function:

```
-[LPMetadataProvider _shouldFailFetchDueToLockdownModeWithCompletionHandler:]
```

Snippet:

```
if [LPTestingOverrides forceEnableLockdownMode] | [LockdownModeManager.shared enabled]
{
  -[LPAssociatedApplicationMetadataFetcher _failedWithErrorCode:underlyingError:]
}
```

NEW dylibs

- /System/Library/Frameworks/LinkPresentation.framework/LinkPresentation
- /System/Library/PrivateFrameworks/WiFiKitUI.framework/WiFiKitUI
- /System/Library/PrivateFrameworks/WiFiKit.framework/WiFiKit

WiFiKit & WiFiKitUI

WiFiKit

WFClient->_lockdownModeEnabled gets set in the function

```
[WFClient initWithCallbackRunLoop:]
```

Where if it is set to true it will also change WFClient->_askToJoinMode from 2 to -1

There are also a series of WFNetworkListController functions that appear to prompt the user when they are about to join an insecure network and tell them why it's insecure allowing them to decide against it.

WiFiKit & WiFiKitUI

WiFiKitUI

```
+[WFLockdownModeAlertController lockdownModeAlertControllerWithNetworkName:securityType:completionHandler:]
```

Called via block invoke in the WiFiKit framework in the function:

```
-[WFNetworkListController _associateToUserSuppliedNetwork:]
```

Also called by the WiFiKit framework function:

```
-[WFNetworkListController _canStartAssociationToNetwork:]
```

```
› plutil -p /System/Library/PrivateFrameworks/WiFiKitUI.framework/WiFiKitUILocalizableStrings.loctable
    "kWFLocLockdownModeCaptiveAlertMessage" => "This network may be unsafe to join in Lockdown Mode."
    "kWFLocLockdownModeCaptiveAlertTitle" => "Join Captive Network “%@”?"
    "kWFLocLockdownModeOpenAlertMessage" => "Open networks provide no security and expose all network traffic. This network is unsafe to join in Lockdown Mode."
    "kWFLocLockdownModeOpenAlertTitle" => "Join Unsecured Network “%@”?"
    "kWFLocLockdownModeOtherAlertMessage" => "This network may be unsafe to join in Lockdown Mode."
    "kWFLocLockdownModeOtherAlertTitle" => "Join Network “%@”?"
    "kWFLocLockdownModePasspointAlertMessage" => "This network may be unsafe to join in Lockdown Mode."
    "kWFLocLockdownModePasspointAlertTitle" => "Join Carrier Network “%@”?"
    "kWFLocLockdownModeWAPIAlertMessage" => "WAPI is not considered secure. This network may be unsafe to join in Lockdown Mode."
    "kWFLocLockdownModeWeakAlertTitle" => "Join Weak Security Network “%@”?"
    "kWFLocLockdownModeWEPAlertMessage" => "WEP is not considered secure. This network may be unsafe to join in Lockdown Mode."
    "kWFLocLockdownModeWPAAlertMessage" => "WPA is not considered secure. This network may be unsafe to join in Lockdown Mode."
    "kWFLocLockdownModeWPATKIPAlertMessage" => "WPA/WPA2 (TKIP) is not considered secure. This network may be unsafe to join in Lockdown Mode.
```

ImageIO

/System/Library/Frameworks/ImageIO.framework/ImageIO

I also noticed in ImageIO that it checks for LDM in the function

```
IIOLockDownModeEnabled
```

Which is called by

```
IIOXPCClient_block_invoke
```

Where it enables the feature decode_images_oop_ldm

Decode images "out of process" while in LDM?

NEW FS bins

- /usr/sbin/wifid
- /Applications/Setup.app/Setup
- /System/Library/Frameworks/CoreTelephony.framework/Support/CommCenter
- /System/Library/PrivateFrameworks/LockdownMode.framework/lockdownmoded

NEW FS bins

- /usr/sbin/wifid
- /Applications/Setup.app/Setup
- /System/Library/Frameworks/CoreTelephony.framework/Support/CommCenter
- /System/Library/PrivateFrameworks/LockdownMode.framework/lockdownmoded

wifid

- Responsible for enforcing the insecure WiFi restrictions which in Lockdown Mode
- For example will prevent certain ssids from waking up the host while in LDM
- Will apply APPLE80211_ASSOC_F_NO_MIXED_WPA_WPA2 while auto joining networks

NEW FS bins

- /usr/sbin/wifid
- /Applications/Setup.app/Setup
- /System/Library/Frameworks/CoreTelephony.framework/Support/CommCenter
- /System/Library/PrivateFrameworks/LockdownMode.framework/lockdownmoded

Setup

Another Swift app

In `-[BuddyLockdownModeController viewDidLoad]` the user is given the option to enable Lockdown Mode with or without restart OR to setup later in Settings.

This is obviously a NEW UI step when setting up a NEW device similar to Screen Time etc.

```
› plutil -p /Applications/Setup.app/en.lproj/Localizable.strings
  "ENABLE_LOCKDOWN_WITH_RESTART" => "Turn On & Restart"
  "ENABLE_LOCKDOWN_WITHOUT_RESTART" => "Turn On"
  <SNIP>
  "LOCKDOWN_MODE_DETAIL_IPAD" => "Lockdown Mode is enabled on another device associated with your Apple account. Would you like to turn it on for this iPad?"
  "LOCKDOWN_MODE_DETAIL_IPHONE" => "Lockdown Mode is enabled on another device associated with your Apple account. Would you like to turn it on for this iPhone?"
  "LOCKDOWN_MODE_DETAIL_IPOD" => "Lockdown Mode is enabled on another device associated with your Apple account. Would you like to turn it on for this iPod touch?"
  "LOCKDOWN_MODE_TITLE" => "Lockdown Mode"
```

NEW FS bins

- /usr/sbin/wifid
- /Applications/Setup.app/Setup
- /System/Library/Frameworks/CoreTelephony.framework/Support/CommCenter
- /System/Library/PrivateFrameworks/LockdownMode.framework/lockdownmoded

CommCenter

A bit of a PitA to RE in IDA Pro and appears to be stripped.

However, it appears that if LDM is enabled it logs the following:

- "#I SMS composition during lockdown”
- "#I Temporary 2G enablement not needed. Triggering E911 state enter"
- "#I Emergency Text: Bypassing home country check"
- "#I Emergency SMS will be released”
- "#I 2G was enabled when in lockedown, resetting the preference”
- "2G was disabled when not in lockedown, resetting the preference"

This is obviously responsible for disabling 2G support when LDM is enabled.

Also allowing 2G for emergency calls when LDM is enabled.

NEW FS bins

- /usr/sbin/wifid
- /Applications/Setup.app/Setup
- /System/Library/Frameworks/CoreTelephony.framework/Support/CommCenter
- /System/Library/PrivateFrameworks/LockdownMode.framework/lockdownmoded

lockdownmoded

- The daemon component of the NEW LDM framework
- Also written in Swift


```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
	<key>application-identifier</key>
	<string>lockdownmoded</string>
	<key>com.apple.developer.ubiquity-kvstore-identifier</key>
	<string>com.apple.LockdownMode</string>
	<key>com.apple.frontboard.shutdown</key>
	<true/>
	<key>com.apple.managedconfiguration.feature-setting.allowLockdownMode</key>
	<true/>
	<key>com.apple.nano.nanoregistry.generalaccess</key>
	<true/>
	<key>com.apple.private.CoreAuthentication.SPI</key>
	<true/>
	<key>com.apple.private.LocalAuthentication.CallerName</key>
	<true/>
	<key>com.apple.private.LocalAuthentication.Storage</key>
	<true/>
	<key>com.apple.private.LocalAuthentication.Storage.SetLockdownMode</key>
	<true/>
	<key>com.apple.private.MobileGestalt.AllowedProtectedKeys</key>
	<array>
		<string>SerialNumber</string>
	</array>
	<key>com.apple.private.accounts.allaccounts</key>
	<true/>
	<key>com.apple.private.admin.writeconfig</key>
	<true/>
	<key>com.apple.private.airdrop.settings</key>
	<true/>
	<key>com.apple.private.applecredentialmanager.allow</key>
	<true/>
	<key>com.apple.private.applecredentialmanager.devicerestrictedmode.allow</key>
	<true/>
	<key>com.apple.private.ids.messaging</key>
	<array>
		<string>com.apple.private.alloy.lockdownmode</string>
	</array>
	<key>com.apple.private.ids.messaging.urgent-priority</key>
	<array>
		<string>com.apple.private.alloy.lockdownmode</string>
	</array>
	<key>com.apple.private.lockdown.reset-pairing</key>
	<true/>
	<key>com.apple.private.sandbox.profile:embedded</key>
	<string>autobox</string>
	<key>com.apple.private.usernotifications.bundle-identifiers</key>
	<array>
		<string>com.apple.LockdownMode</string>
	</array>
	<key>com.apple.private.xpc.launchd.reboot</key>
	<true/>
	<key>com.apple.security.exception.files.absolute-path.read-only</key>
	<array>
		<string>/private/var/</string>
	</array>
	<key>com.apple.security.exception.files.absolute-path.read-write</key>
	<string>/private/var/mobile/tmp/com.apple.lockdownmoded/</string>
	<key>com.apple.security.exception.files.home-relative-path.read-write</key>
	<array>
		<string>/Library/LockdownMode/</string>
	</array>
	<key>com.apple.security.exception.process-info</key>
	<true/>
	<key>com.apple.security.exception.shared-preference.read-only</key>
	<array>
		<string>com.apple.purplebuddy</string>
	</array>
	<key>com.apple.security.exception.shared-preference.read-write</key>
	<array>
		<string>com.apple.airplay</string>
		<string>com.apple.Sharing</string>
		<string>com.apple.sharingd</string>
		<string>com.apple.coremedia</string>
		<string>com.apple.controlcenter</string>
		<string>kCFPreferencesAnyApplication</string>
	</array>
	<key>com.apple.security.exception.sysctl.read-only</key>
	<array>
		<string>security.mac.lockdown_mode_state</string>
	</array>
	<key>com.apple.springboard.CFUserNotification</key>
	<true/>
	<key>com.apple.telephonyutilities.callservicesd</key>
	<array>
		<string>access-calls</string>
	</array>
	<key>platform-application</key>
	<true/>
</dict>
</plist>
```

lockdownmoded

```
› ipsw macho info /System/Library/PrivateFrameworks/LockdownMode.framework/lockdownmoded --swift | grep class
    class symbolic lockdownmoded.LockdownModeDaemon {
    class symbolic lockdownmoded.LockdownModePhotos {}
    class symbolic lockdownmoded.LockdownModeSharing {}
    objc class symbolic lockdownmoded.NotificationsManager._$sSo8NSObjectC {
    class symbolic lockdownmoded.LockdownModeAMFI {}
    class symbolic lockdownmoded.LockdownModeManagedConfiguration {}
    objc class symbolic lockdownmoded.LockdownModeServer._$sSo8NSObjectC {
    class symbolic lockdownmoded.LockdownModeACM {}
    class symbolic lockdownmoded.RestrictionsManager {
    class symbolic lockdownmoded.LockdownModeCoreMedia {}
    class symbolic lockdownmoded.AnalyticsManager {}
    class symbolic lockdownmoded.LockdownModeSafari {}
    class symbolic lockdownmoded.Restriction {
    objc class symbolic lockdownmoded.IDSManager._$sSo8NSObjectC
```

```
enum symbolic lockdownmoded.RestrictionsManager.RestrictionKey {
        let airPlayToTV
        let linkPreviews
        let trm
        let sharedAlbums
        let airPlayReceiver
        let airDropAutoAccept
        let airDropDiscoverableMode
        let developerMode
}
```

lockdownmoded

What can talk to it?

```
> i
```

LockdownMode FW

```
› ipsw dyld macho dyld_shared_cache_arm64e LockdownMode --objc -V
```

LockdownModeXNU ?!?

```
@protocol LockdownMode.LockdownModeProtocol  // 0x230c70ea8
/* instance methods */
- (void)migrateIfNeededWithCompletion:(id /* block */)arg1;
- (void)enableIfNeededWithReboot:(BOOL)arg1 completion:(id /* block */)arg2;
- (void)getEnabledInAccountWithSynchronize:(BOOL)arg1 completion:(id /* block */)arg2;
- (void)notifyRestrictionChanged:(id)arg1 completion:(id /* block */)arg2;
- (void)rebootIfNeededWithCompletion:(id /* block */)arg1;
- (void)setEnabledWithEnabled:(BOOL)arg1 options:(id)arg2 completion:(id /* block */)arg3;
@end

@interface LockdownModeManager : NSObject {
  /* instance variables */                    // +size   offset
  LockdownModeManagerInternal *_underlyingObject; // +0x8    0x8
}

@property (readonly) LockdownModeManagerInternal *underlyingObject;
@property (readonly, nonatomic) BOOL enabled;
@property (readonly, nonatomic) BOOL enabledInAccount;

+ (id)shared;

- (id)init;
- (BOOL)enabled;
- (id)underlyingObject;
- (void).cxx_destruct;
- (void)migrateIfNeededWithCompletion:(id /* block */)arg1;
- (BOOL)enabledInAccount;
- (void)setEnabled:(BOOL)arg1 options:(id)arg2 completion:(id /* block */)arg3;
- (void)getEnabledInAccount:(BOOL)arg1 completion:(id /* block */)arg2;
- (void)enableIfNeeded:(BOOL)arg1 completion:(id /* block */)arg2;
- (void)notifyRestrictionChanged:(id)arg1;
- (void)rebootIfNeeded;
@end

@interface LockdownMode.LockdownModeXNU : Swift._SwiftObject { // 0x22f152888 (Swift)
@end
```

Please don’t

spoiler alert… they did

```
› ipsw macho info kernelcache.release.Mac14,5_6_8_9_10_12_13_14 --fileset-entry kernel --symbols --strings | grep -i
lockdown_mode

  0xfffffe0008a6aff8:  (__TEXT_EXEC,__text) external     _disable_lockdown_mode
  0xfffffe0008a6afd8:  (__TEXT_EXEC,__text) external     _enable_lockdown_mode
  0xfffffe0008a6af00:  (__TEXT_EXEC,__text) external     _get_lockdown_mode_state
  0xfffffe00070b2910: "lockdown_mode_state"
  0xfffffe00070b2938: "lockdown_mode: error getting state from nvram\n"
  0xfffffe00070b2967: "lockdown_mode: lockdown mode in nvram is %s\n"
  0xfffffe00070b299b: “get_lockdown_mode_state"

› ipsw macho info kernelcache.release.Mac                                    -i lockdown

  099: LC_FILESET_ENTRY offset=0x0003                                    m.apple.driver.AppleLockdownMode
```

LDM

Now in XNU

How Bad Is It?

```
> i
```

How Bad Is It?

not that bad 🥺

In the KEXT com.apple.driver.AppleLockdownMode

```
bool AppleLockdownMode::start(IOService *a1, IOService *a2)
{
  v2 = IOService::start(a1, a2);
  if ( v2 )
    IOLog("LockdownMode: started (no-op)\n”);
  else
    IOLog("LockdownMode: failed to start");
  return v2;
}
```

Appears to be a no-op for now

In the KEXT com.apple.driver.AppleCredentialManager

```
ACMLockdownModeKernelService::commandProcessedGated(acm_
unsigned char const*, unsigned int)
```

This function calls via kernel_thread_start enable_lockdown_mode or disable_
input struct and is called from AppleCredentialManager::performCommandGa

In the KEXT com.apple.driver.AppleBCMWLANCore

```
AppleBCMWLANJoinAdapter::performJoin(apple80211_assoc_candida
```

“[ik] %s@%d:%s: Disable roaming when joining an RSN_AKM_PSK network in lockd

… for now

🔑 Takeaways

- Very interesting to see what Apple perceives as their attack surface
- LDM very much work in progress and will continue to grow
- We need to get good at reversing Swift like… right now
- `ipsw` is great! You should use it for more than just downloading IPSWs ;)

Thank you! ❤️

- github.com/blacktop/ipsw
- blacktop@proton.me

Oh yeah, we're hiring ;)