appdome

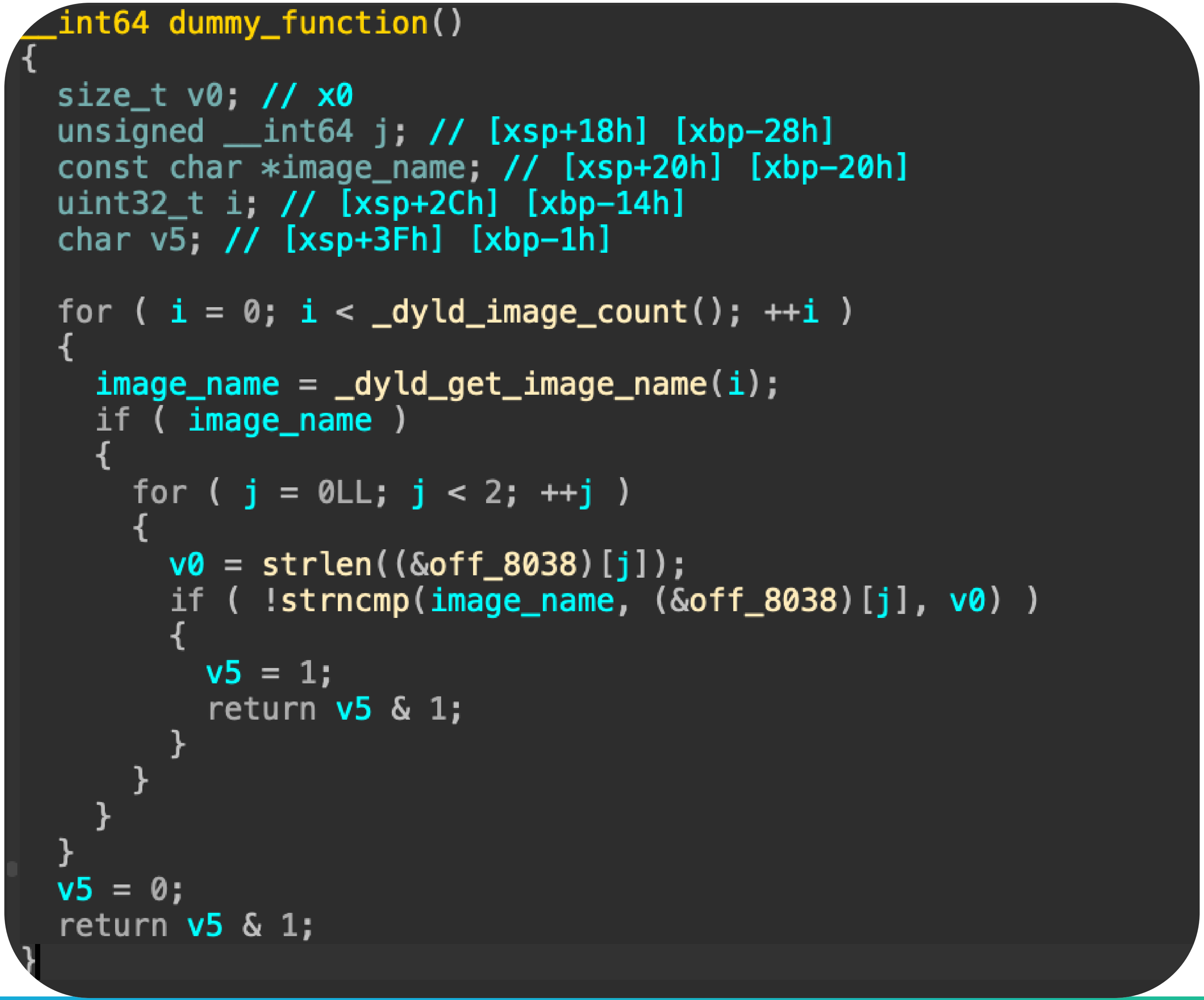
Where DYLD you hide?

Leveraging the Mach-O Format and the iOS Dynamic Linker

For Advanced Injection Techniques and Research

Barak Aharoni | Appdome

2024



```
__int64 dummy_function()
{
  size_t v0; // x0
  unsigned __int64 j; // [xsp+18h] [xbp-28h]
  const char *image_name; // [xsp+20h] [xbp-20h]
  uint32_t i; // [xsp+2Ch] [xbp-14h]
  char v5; // [xsp+3Fh] [xbp-1h]

  for ( i = 0; i < _dyld_image_count(); ++i )
  {
    image_name = _dyld_get_image_name(i);
    if ( image_name )
    {
      for ( j = 0LL; j < 2; ++j )
      {
        v0 = strlen((&off_8038)[j]);
        if ( !strncmp(image_name, (&off_8038)[j], v0) )
        {
          v5 = 1;
          return v5 & 1;
        }
      }
    }
  }
  v5 = 0;
  return v5 & 1;
}
```

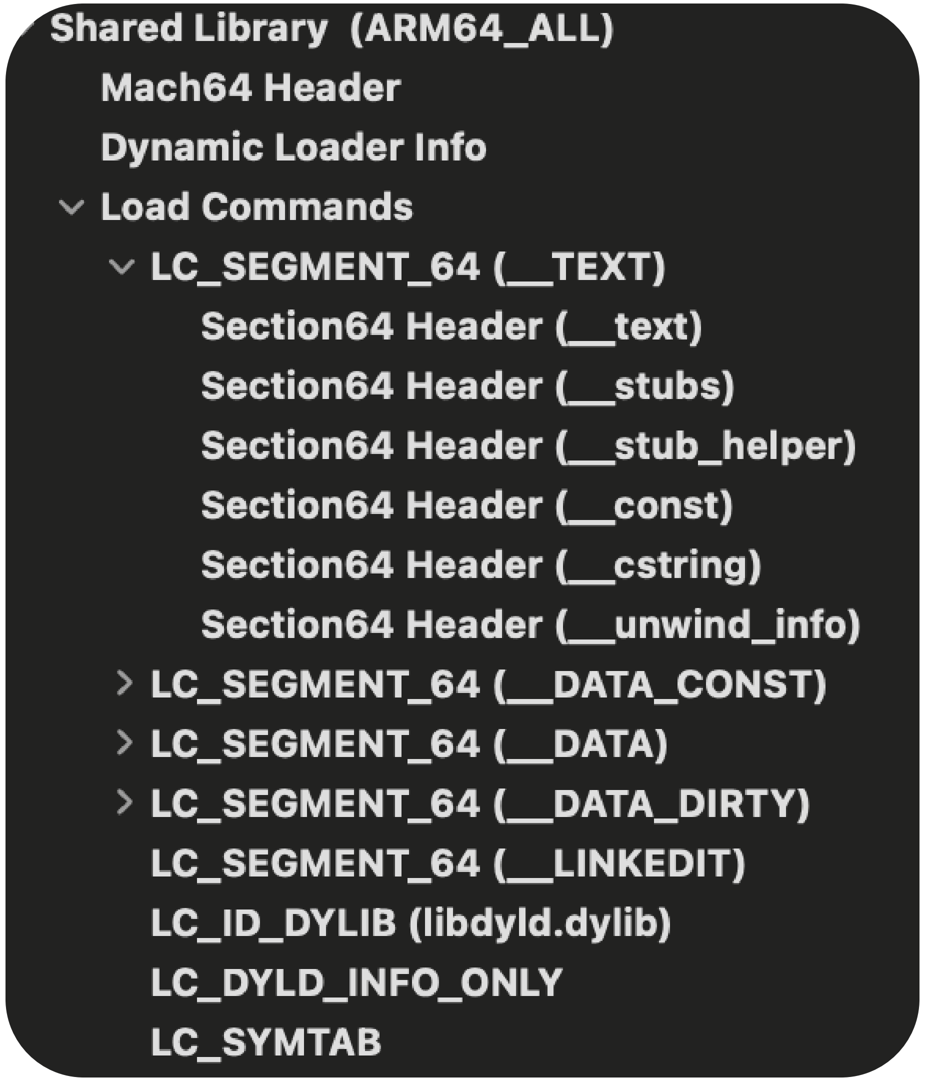
Agenda

- Background
- Utilized DYLD’s Behavior for Malicious Purposes
- Debugging and Research Methodologies
- Mitigations and Recommendations

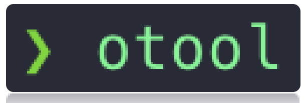
Background

Mach-O Format

- Mach Header
- Load Commands
  - LC_SEGMENT
  - LC_LOAD*_DYLIB
  - LC_MAIN
- Segments
  - __TEXT
  - __DATA
  - __LINKEDIT
- Sections
  - __text
  - __data
  - __mod_init_func



```
Shared Library  (ARM64_ALL)
    Mach64 Header
    Dynamic Loader Info
  ⌄ Load Commands
    ⌄ LC_SEGMENT_64 (__TEXT)
        Section64 Header (__text)
        Section64 Header (__stubs)
        Section64 Header (__stub_helper)
        Section64 Header (__const)
        Section64 Header (__cstring)
        Section64 Header (__unwind_info)
    › LC_SEGMENT_64 (__DATA_CONST)
    › LC_SEGMENT_64 (__DATA)
    › LC_SEGMENT_64 (__DATA_DIRTY)
      LC_SEGMENT_64 (__LINKEDIT)
      LC_ID_DYLIB (libdyld.dylib)
      LC_DYLD_INFO_ONLY
      LC_SYMTAB
```



What Happens When You Click an Application?

iOS Dynamic Linker (DYLD)

Load	Dyld loads the executable and libraries
Rebase	Adjusts memory addresses
Bind	Resolves and links symbols
Initialize	Calls initialization functions and constructors
Execute	Transfers control to the app's entry point

iOS Dynamic Linker (DYLD)

MH_DYLINKER

dyld

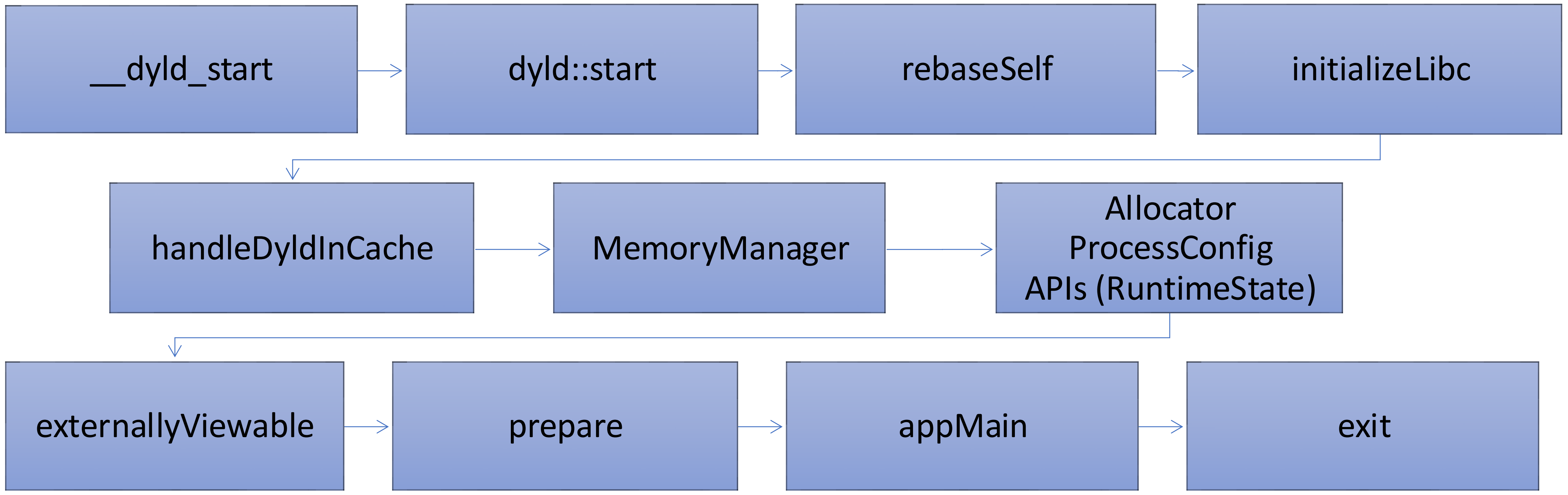
The dynamic linker that loads and links dynamic libraries before app launch

MH_DYLIB

libdyld.dylib

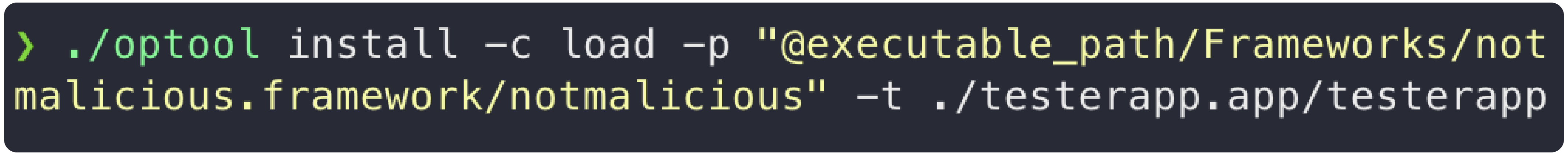
Provides APIs to interact with dyld at runtime

dyldMain.cpp start()

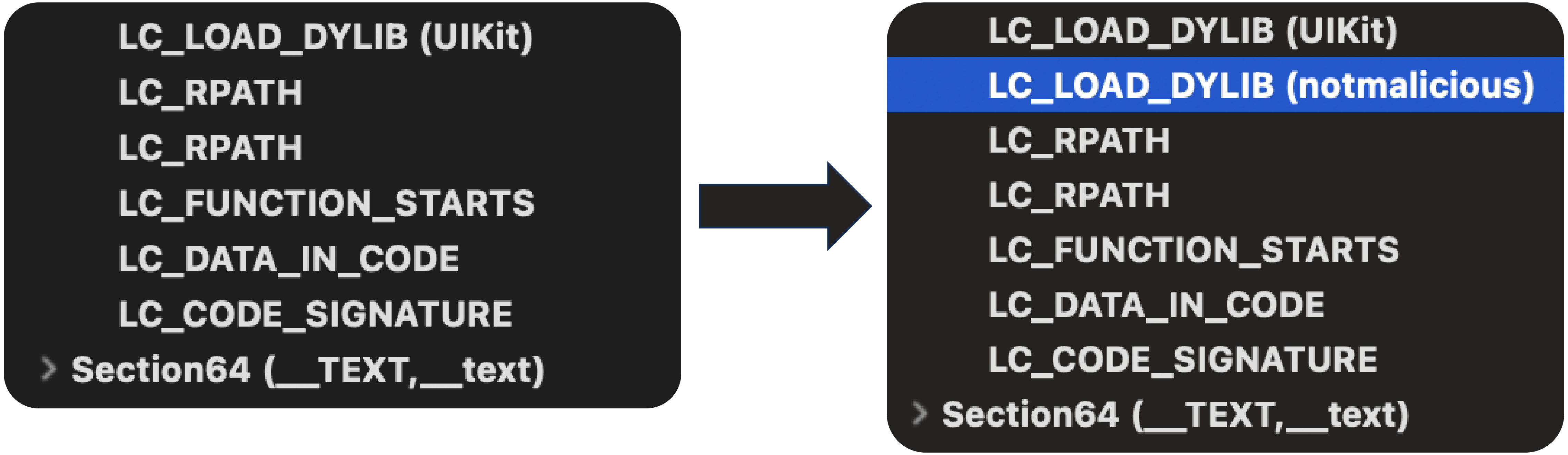


Utilized DYLD’s Behavior for Malicious Purposes

Static Load Command Injection



```
❯ ./optool install -c load -p "@executable_path/Frameworks/not
malicious.framework/notmalicious" -t ./testerapp.app/testerapp
```



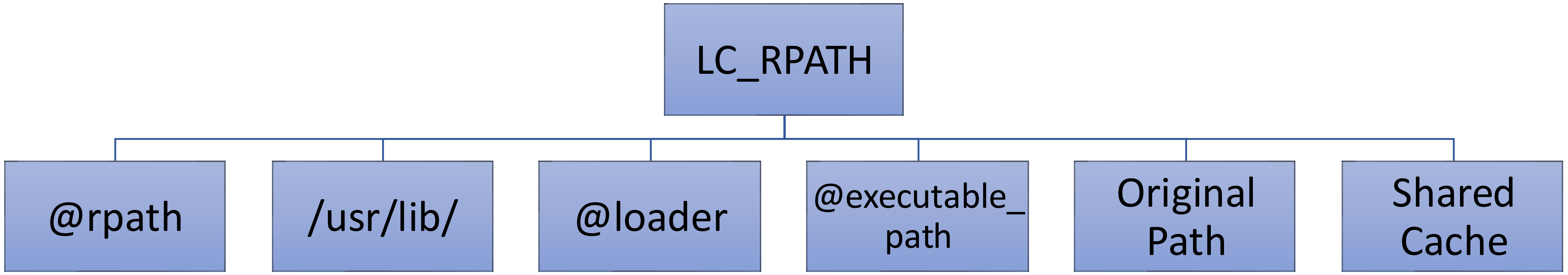
```
LC_LOAD_DYLIB (UIKit)
LC_RPATH
LC_RPATH
LC_FUNCTION_STARTS
LC_DATA_IN_CODE
LC_CODE_SIGNATURE
› Section64 (__TEXT,__text)
```

```
LC_LOAD_DYLIB (UIKit)
LC_LOAD_DYLIB (notmalicious)
LC_RPATH
LC_RPATH
LC_FUNCTION_STARTS
LC_DATA_IN_CODE
LC_CODE_SIGNATURE
› Section64 (__TEXT,__text)
```

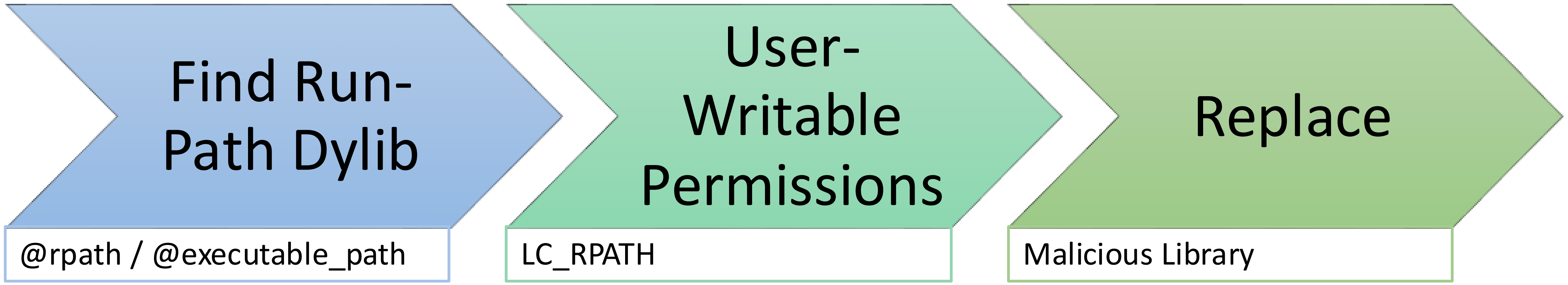
DYLD Search Path Hierarchy

- **LC_RPATH** specifies a runtime search path for locating dynamically linked libraries
- **DYLD_*_PATH** environment variables

- **getLoader()** in **dyldMain`prepare()**



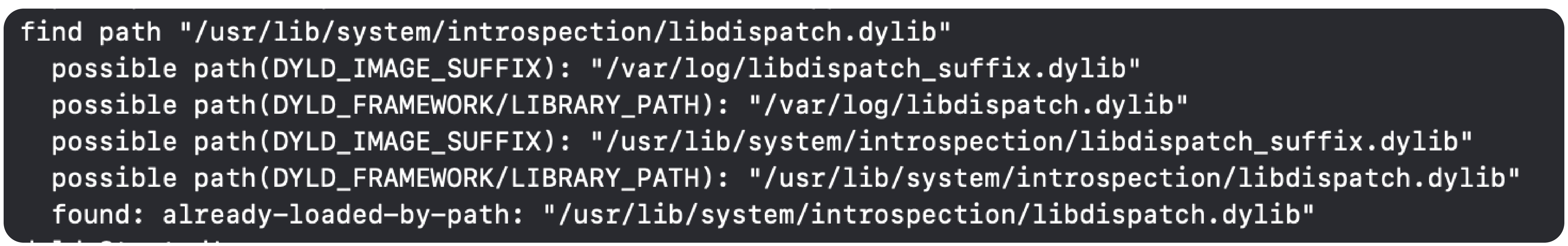
Dylib Hijacking



Dynamic Dylib Hijacking

- Set Path Related Environment Variables
    - DYLD_IMAGE_SUFFIX
    - DYLD_FRAMEWORK_PATH
    - DYLD_LIBRARY_PATH
    - DYLD_FALLBACK_FRAMEWORK_PATH
    - DYLD_FALLBACK_LIBRARY_PATH

DyldProcessConfig`PathOverrides

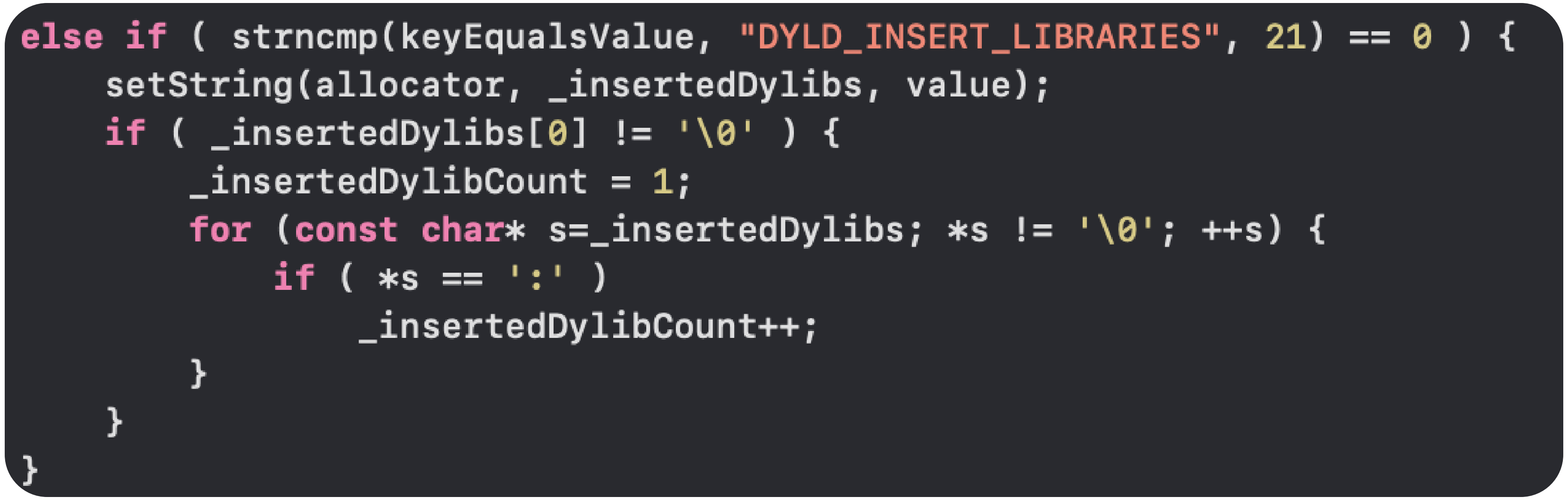


```
find path "/usr/lib/system/introspection/libdispatch.dylib"
  possible path(DYLD_IMAGE_SUFFIX): "/var/log/libdispatch_suffix.dylib"
  possible path(DYLD_FRAMEWORK/LIBRARY_PATH): "/var/log/libdispatch.dylib"
  possible path(DYLD_IMAGE_SUFFIX): "/usr/lib/system/introspection/libdispatch_suffix.dylib"
  possible path(DYLD_FRAMEWORK/LIBRARY_PATH): "/usr/lib/system/introspection/libdispatch.dylib"
  found: already-loaded-by-path: "/usr/lib/system/introspection/libdispatch.dylib"
```

Dynamic Injection

- Use **DYLD_INSERT_LIBRARIES** environment variable to load a library before other dylibs

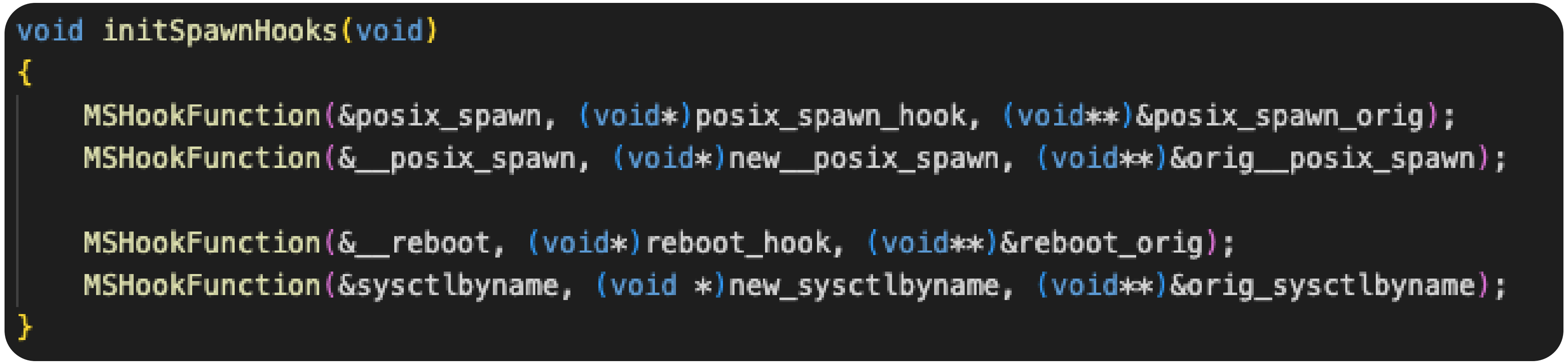
dyldMain`DyldProcessConfig.cpp



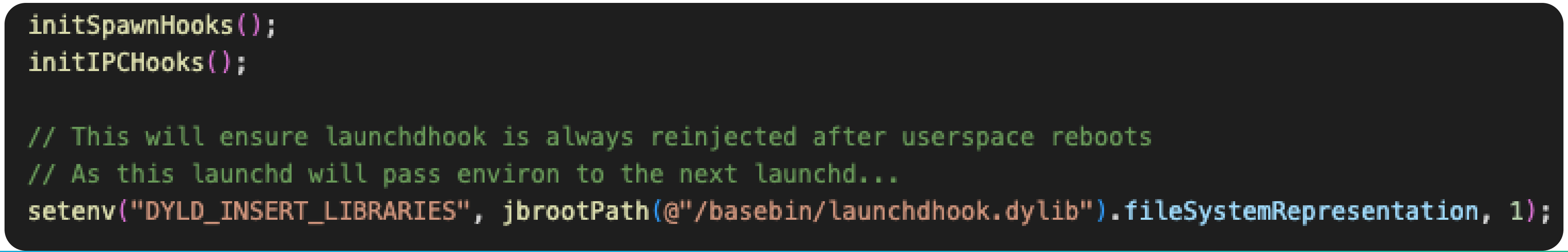
```
else if ( strncmp(keyEqualsValue, "DYLD_INSERT_LIBRARIES", 21) == 0 ) {
    setString(allocator, _insertedDylibs, value);
    if ( _insertedDylibs[0] != '\0' ) {
        _insertedDylibCount = 1;
        for (const char* s=_insertedDylibs; *s != '\0'; ++s) {
            if ( *s == ':' )
                _insertedDylibCount++;
        }
    }
}
```

Dynamic Injection – Dopamine (RootHide)

- Set **DYLD_INSERT_LIBRARIES** environment variable to inject **launchdhook.dylib**
- initSpawnHooks use Method Swizzling on posix_spawn



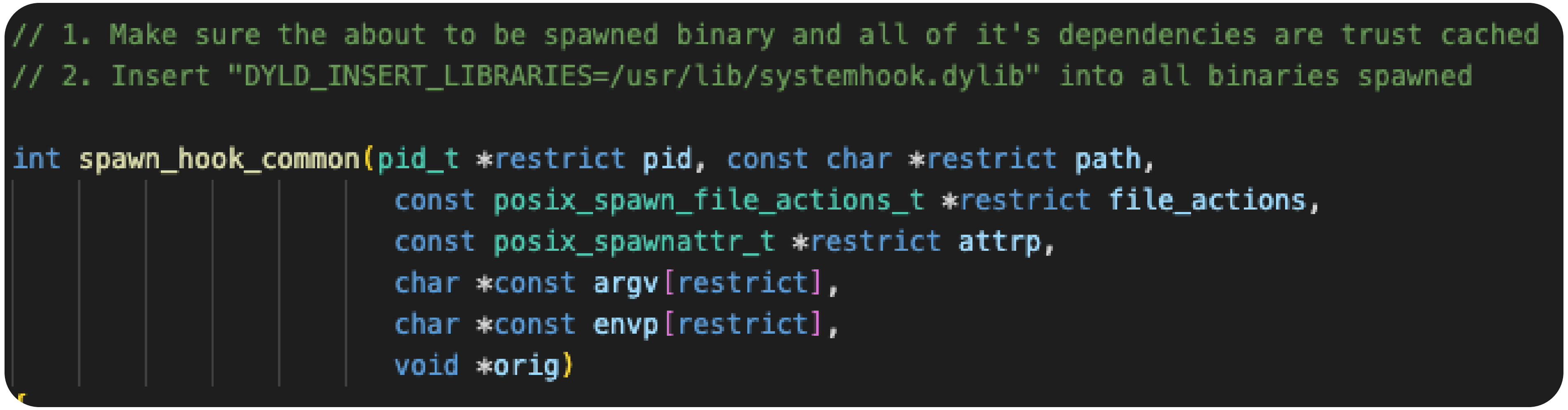
```
void initSpawnHooks(void)
{
    MSHookFunction(&posix_spawn, (void*)posix_spawn_hook, (void**)&posix_spawn_orig);
    MSHookFunction(&__posix_spawn, (void*)new__posix_spawn, (void**)&orig__posix_spawn);

    MSHookFunction(&__reboot, (void*)reboot_hook, (void**)&reboot_orig);
    MSHookFunction(&sysctlbyname, (void *)new_sysctlbyname, (void**)&orig_sysctlbyname);
}
```



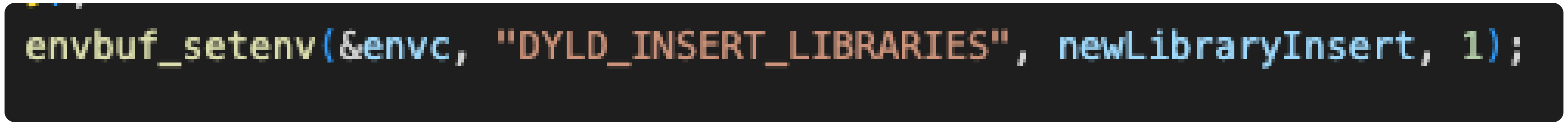
```
initSpawnHooks();
initIPCHooks();

// This will ensure launchdhook is always reinjected after userspace reboots
// As this launchd will pass environ to the next launchd...
setenv("DYLD_INSERT_LIBRARIES", jbrootPath(@"/basebin/launchdhook.dylib").fileSystemRepresentation, 1);
```

Dynamic Injection – Dopamine (RootHide)

- **spawn_hook_common** inserts **systemhook.dylib** into all binaries spawned
- Set **DYLD_INSERT_LIBRARIES** environment variable



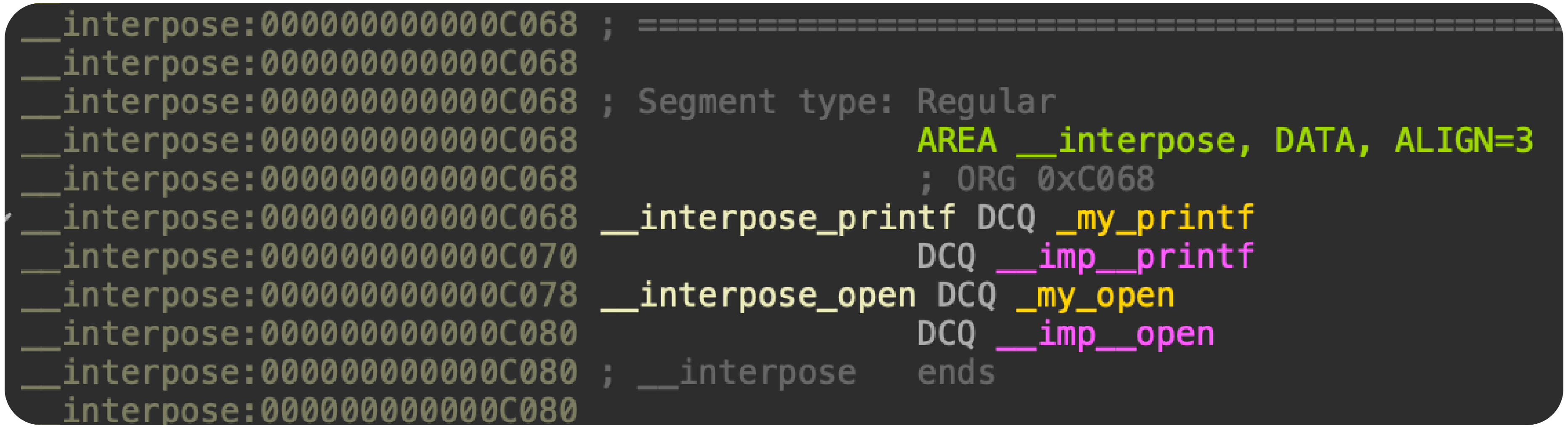
```
// 1. Make sure the about to be spawned binary and all of it's dependencies are trust cached
// 2. Insert "DYLD_INSERT_LIBRARIES=/usr/lib/systemhook.dylib" into all binaries spawned

int spawn_hook_common(pid_t *restrict pid, const char *restrict path,
                      const posix_spawn_file_actions_t *restrict file_actions,
                      const posix_spawnattr_t *restrict attrp,
                      char *const argv[restrict],
                      char *const envp[restrict],
                      void *orig)
```



```
envbuf_setenv(&envc, "DYLD_INSERT_LIBRARIES", newLibraryInsert, 1);
```

Interposing

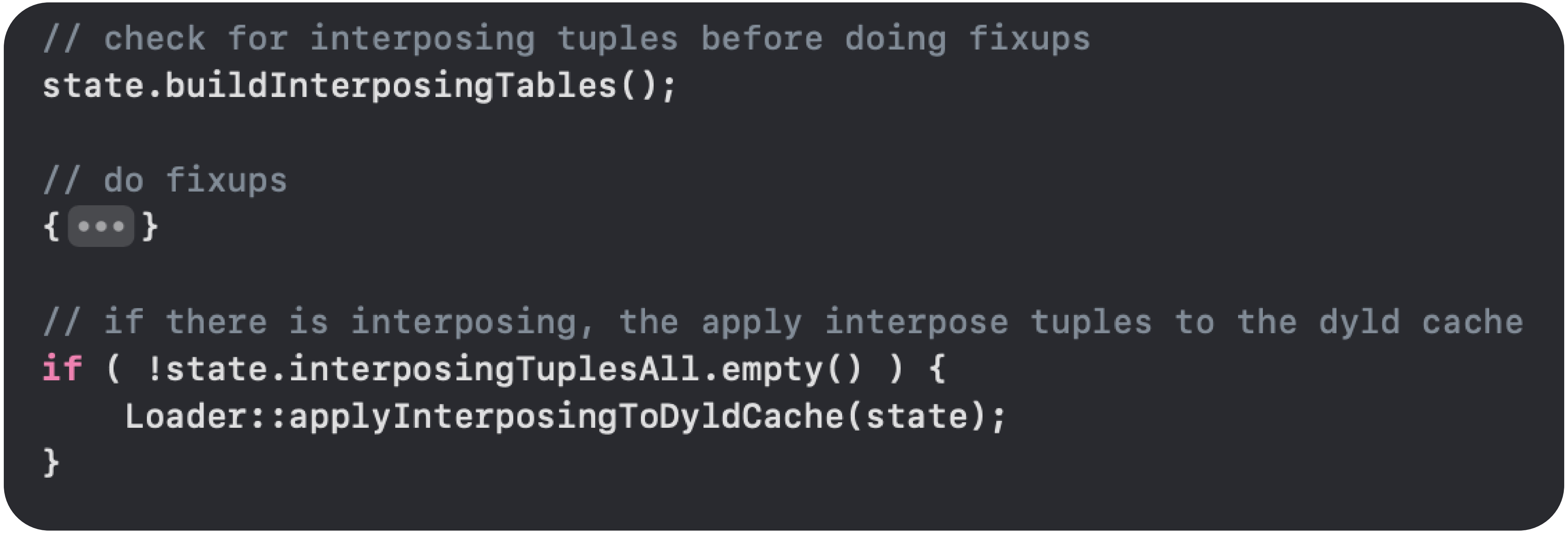
```
// From mach-o/dyld-interposing.h
#define DYLD_INTERPOSE(_replacement,_replacee) \
   __attribute__((used)) static struct{ const void* replacement; const void* replacee; }
       _interpose_##_replacee \
            __attribute__ ((section ("__DATA,__interpose"))) = { (const void*)(unsigned
                long)&_replacement, (const void*)(unsigned long)&_replacee };

DYLD_INTERPOSE(my_printf, printf);
DYLD_INTERPOSE(my_open, open);
```



```
__interpose:000000000000C068 ; ===================================================
__interpose:000000000000C068
__interpose:000000000000C068 ; Segment type: Regular
__interpose:000000000000C068                 AREA __interpose, DATA, ALIGN=3
__interpose:000000000000C068                 ; ORG 0xC068
__interpose:000000000000C068 __interpose_printf DCQ _my_printf
__interpose:000000000000C070                 DCQ __imp__printf
__interpose:000000000000C078 __interpose_open DCQ _my_open
__interpose:000000000000C080                 DCQ __imp__open
__interpose:000000000000C080 ; __interpose   ends
__interpose:000000000000C080
```

Interposing

dyldMain`prepare

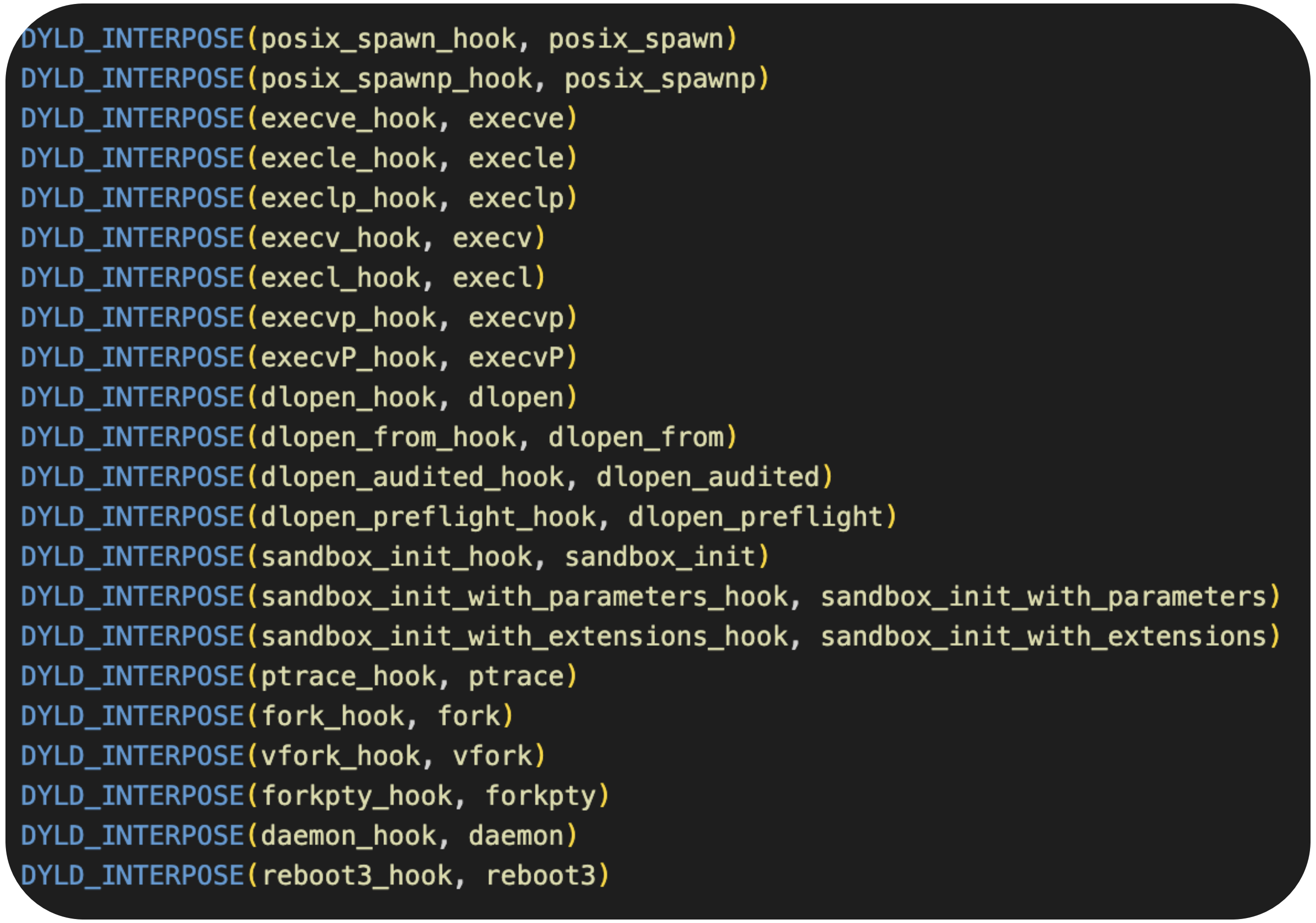


```
// check for interposing tuples before doing fixups
state.buildInterposingTables();

// do fixups
{ ••• }

// if there is interposing, the apply interpose tuples to the dyld cache
if ( !state.interposingTuplesAll.empty() ) {
    Loader::applyInterposingToDyldCache(state);
}
```

Interposing – Dopamine (RootHide)

- **execve_hook** – Controls process launches
- **open_hook** – Manages file access
- **dlopen_hook** – Restricts library loading
- **ptrace_hook** – apply debug flags to when needed
- **sandbox_init** - Enforces sandbox policies

systemhook.dylib – BaseBin/systemhook/src/main.c

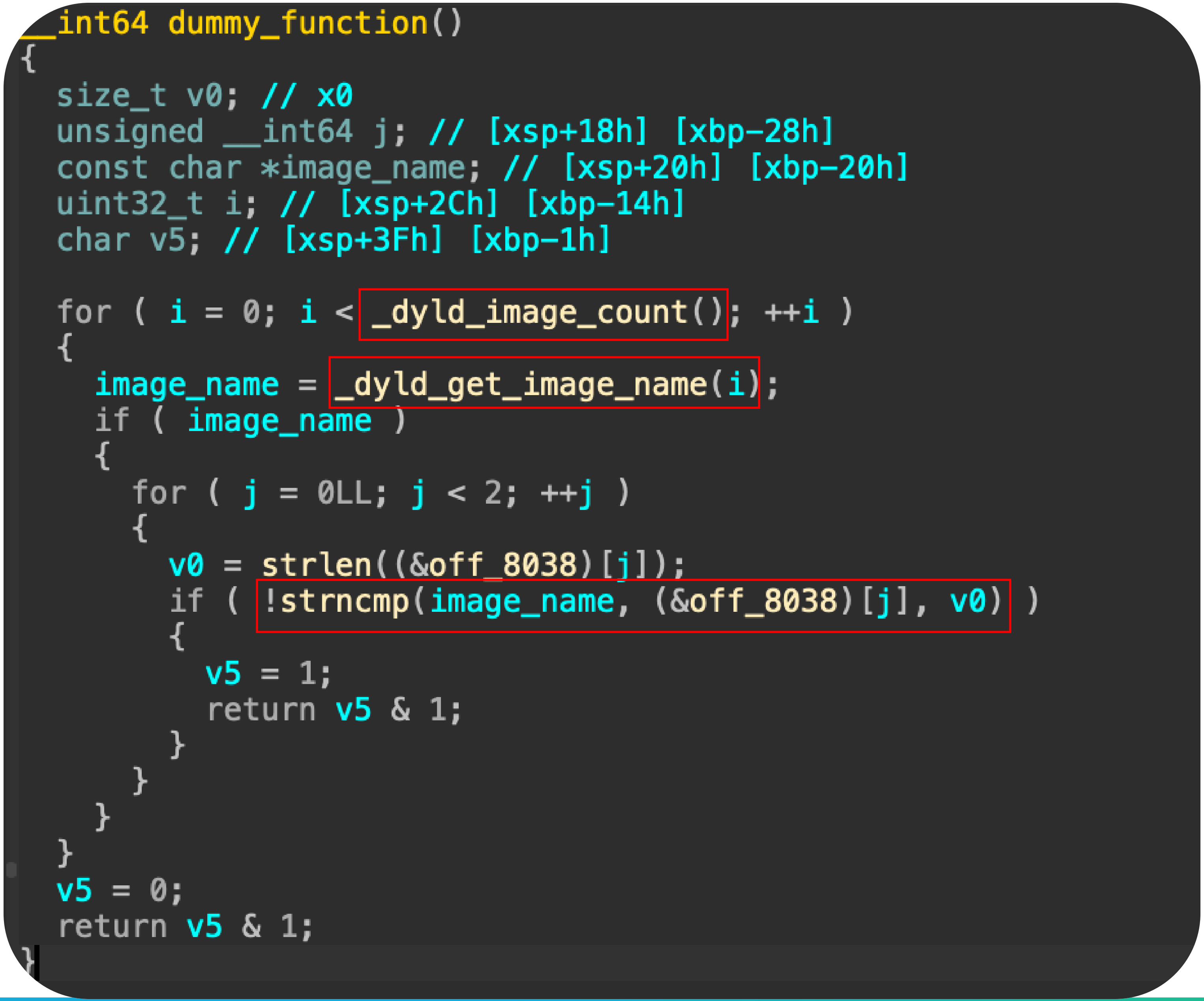


```
DYLD_INTERPOSE(posix_spawn_hook, posix_spawn)
DYLD_INTERPOSE(posix_spawnp_hook, posix_spawnp)
DYLD_INTERPOSE(execve_hook, execve)
DYLD_INTERPOSE(execle_hook, execle)
DYLD_INTERPOSE(execlp_hook, execlp)
DYLD_INTERPOSE(execv_hook, execv)
DYLD_INTERPOSE(execl_hook, execl)
DYLD_INTERPOSE(execvp_hook, execvp)
DYLD_INTERPOSE(execvP_hook, execvP)
DYLD_INTERPOSE(dlopen_hook, dlopen)
DYLD_INTERPOSE(dlopen_from_hook, dlopen_from)
DYLD_INTERPOSE(dlopen_audited_hook, dlopen_audited)
DYLD_INTERPOSE(dlopen_preflight_hook, dlopen_preflight)
DYLD_INTERPOSE(sandbox_init_hook, sandbox_init)
DYLD_INTERPOSE(sandbox_init_with_parameters_hook, sandbox_init_with_parameters)
DYLD_INTERPOSE(sandbox_init_with_extensions_hook, sandbox_init_with_extensions)
DYLD_INTERPOSE(ptrace_hook, ptrace)
DYLD_INTERPOSE(fork_hook, fork)
DYLD_INTERPOSE(vfork_hook, vfork)
DYLD_INTERPOSE(forkpty_hook, forkpty)
DYLD_INTERPOSE(daemon_hook, daemon)
DYLD_INTERPOSE(reboot3_hook, reboot3)
```

appdome

Useful DYLD API Functions

- **_dyld_image_count(*void*)** returns the number of loaded images
- **_dyld_get_image_name(*index*)** returns the name of a given library index
- **_dyld_get_image_header(*index*)** returns the base address of a given library index
- **_dyld_register_func_for_add_image(*callback*)** allows to install callbacks which will be called by dyld whenever an image is loaded or unloaded


```
__int64 dummy_function()
{
  size_t v0; // x0
  unsigned __int64 j; // [xsp+18h] [xbp-28h]
  const char *image_name; // [xsp+20h] [xbp-20h]
  uint32_t i; // [xsp+2Ch] [xbp-14h]
  char v5; // [xsp+3Fh] [xbp-1h]

  for ( i = 0; i < _dyld_image_count(); ++i )
  {
    image_name = _dyld_get_image_name(i);
    if ( image_name )
    {
      for ( j = 0LL; j < 2; ++j )
      {
        v0 = strlen((&off_8038)[j]);
        if ( !strncmp(image_name, (&off_8038)[j], v0) )
        {
          v5 = 1;
          return v5 & 1;
        }
      }
    }
  }
  v5 = 0;
  return v5 & 1;
}
```

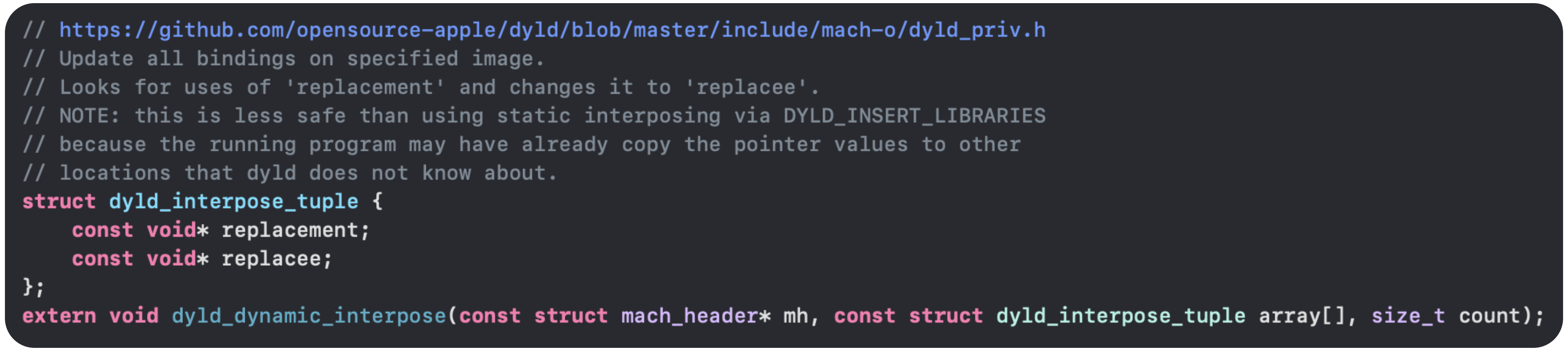
How Malicious Dylibs Can Avoid Detection

```
// Return 1 as the image count
uint32_t _my_dyld_image_count(void)
{
    uint32_t orig_count = _dyld_image_count();
    uint32_t ret = 1;
    printf("Change image_count from: %d to: %d\n", orig_count, ret);
    return ret;
}

// Return only the first image name
const char *_my_dyld_get_image_name(uint32_t image_index)
{
    const char *orig_name = _dyld_get_image_name(image_index);
    const char *ret = _dyld_get_image_name(0);
    printf("Change image_name from: %s to: %s\n", orig_name, ret);
    return ret;
}
```

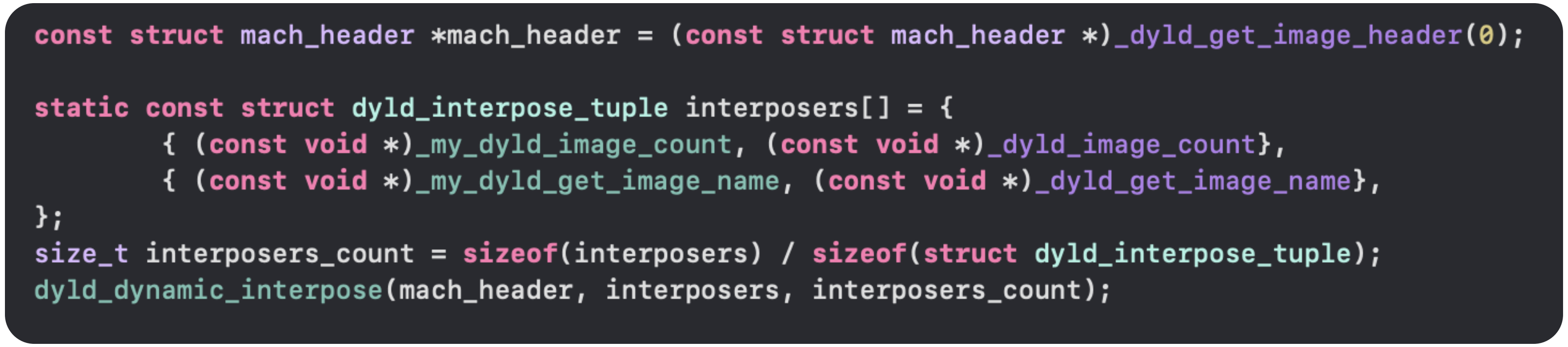
```
DYLD_INTERPOSE(_my_dyld_image_count, _dyld_image_count);
DYLD_INTERPOSE(_my_dyld_get_image_name, _dyld_get_image_name);
```

Dynamic Interposing

- Use **_dyld_dynamic_interpose** to hide libraries from _dyld_* APIs.



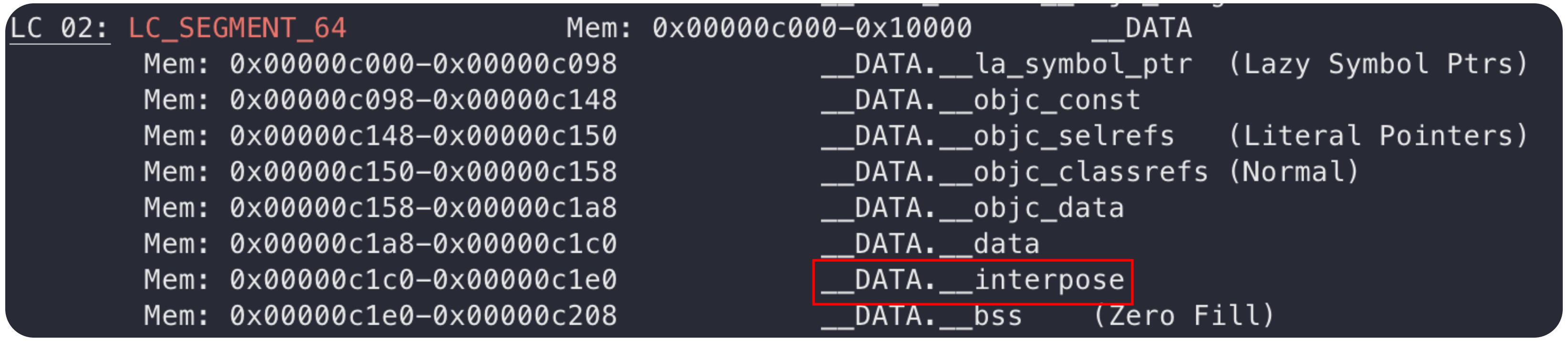
```
// https://github.com/opensource-apple/dyld/blob/master/include/mach-o/dyld_priv.h
// Update all bindings on specified image.
// Looks for uses of 'replacement' and changes it to 'replacee'.
// NOTE: this is less safe than using static interposing via DYLD_INSERT_LIBRARIES
// because the running program may have already copy the pointer values to other
// locations that dyld does not know about.
struct dyld_interpose_tuple {
    const void* replacement;
    const void* replacee;
};
extern void dyld_dynamic_interpose(const struct mach_header* mh, const struct dyld_interpose_tuple array[], size_t count);
```



```
const struct mach_header *mach_header = (const struct mach_header *)_dyld_get_image_header(0);

static const struct dyld_interpose_tuple interposers[] = {
        { (const void *)_my_dyld_image_count, (const void *)_dyld_image_count},
        { (const void *)_my_dyld_get_image_name, (const void *)_dyld_get_image_name},
};
size_t interposers_count = sizeof(interposers) / sizeof(struct dyld_interpose_tuple);
dyld_dynamic_interpose(mach_header, interposers, interposers_count);
```

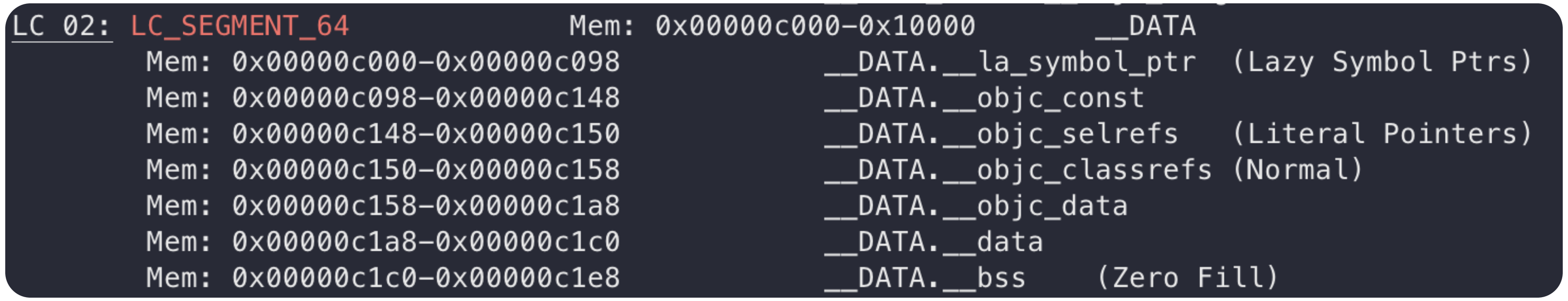
Dynamic Interposing

- No related section in the binary
- Static Interposing



```
LC 02: LC_SEGMENT_64              Mem: 0x00000c000-0x10000        __DATA
        Mem: 0x00000c000-0x00000c098                __DATA.__la_symbol_ptr  (Lazy Symbol Ptrs)
        Mem: 0x00000c098-0x00000c148                __DATA.__objc_const
        Mem: 0x00000c148-0x00000c150                __DATA.__objc_selrefs   (Literal Pointers)
        Mem: 0x00000c150-0x00000c158                __DATA.__objc_classrefs (Normal)
        Mem: 0x00000c158-0x00000c1a8                __DATA.__objc_data
        Mem: 0x00000c1a8-0x00000c1c0                __DATA.__data
        Mem: 0x00000c1c0-0x00000c1e0                __DATA.__interpose
        Mem: 0x00000c1e0-0x00000c208                __DATA.__bss    (Zero Fill)
```

- Dynamic Interposing



```
LC 02: LC_SEGMENT_64              Mem: 0x00000c000-0x10000        __DATA
        Mem: 0x00000c000-0x00000c098                __DATA.__la_symbol_ptr  (Lazy Symbol Ptrs)
        Mem: 0x00000c098-0x00000c148                __DATA.__objc_const
        Mem: 0x00000c148-0x00000c150                __DATA.__objc_selrefs   (Literal Pointers)
        Mem: 0x00000c150-0x00000c158                __DATA.__objc_classrefs (Normal)
        Mem: 0x00000c158-0x00000c1a8                __DATA.__objc_data
        Mem: 0x00000c1a8-0x00000c1c0                __DATA.__data
        Mem: 0x00000c1c0-0x00000c1e8                __DATA.__bss    (Zero Fill)
```

Defensive Position
Debugging and Research Methodologies

Debugging using Environment Variables

DYLD_PRINT_LIBRARIES

DYLD_PRINT_LOADERS

DYLD_PRINT_INITIALIZERS

DYLD_PRINT_SEGMENTS

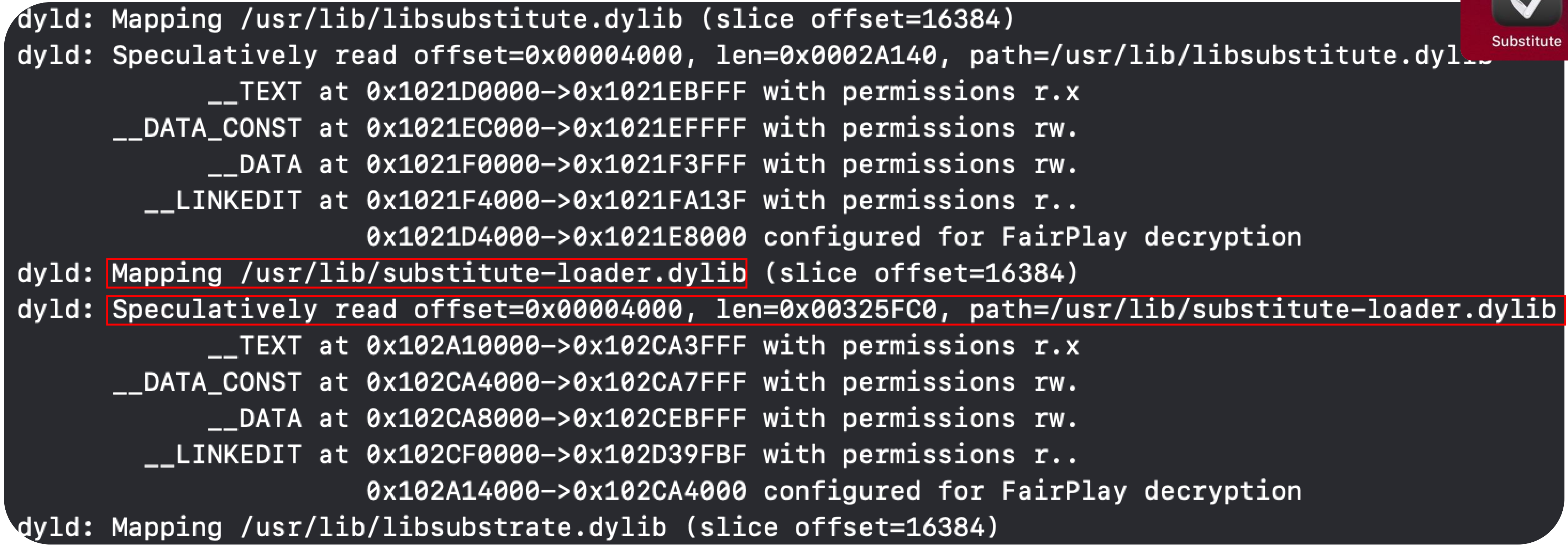
DYLD_PRINT_SEARCHING

DYLD_PRINT_APIS

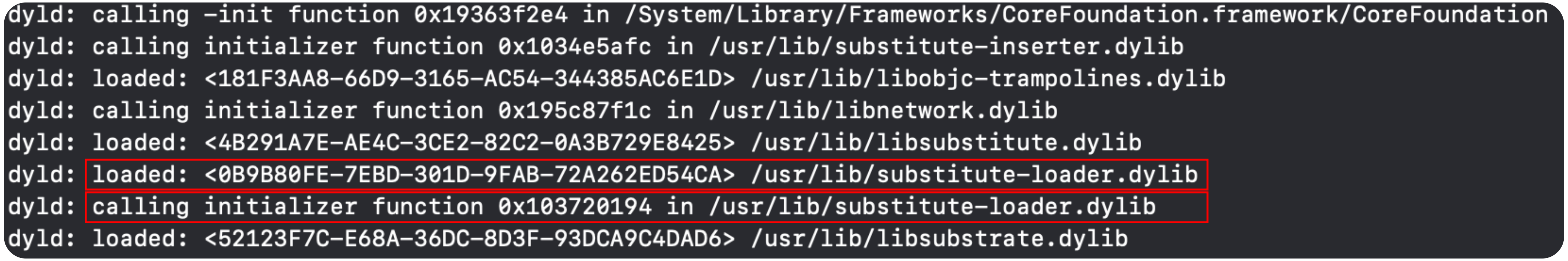
DYLD_PRINT_BINDINGS

DYLD_PRINT_INTERPOSING

Substitute

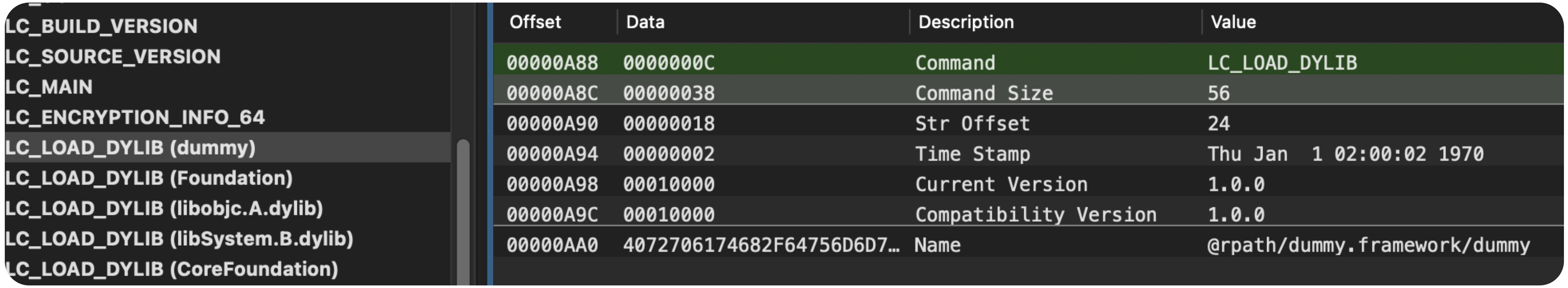


```
dyld: Mapping /usr/lib/libsubstitute.dylib (slice offset=16384)
dyld: Speculatively read offset=0x00004000, len=0x0002A140, path=/usr/lib/libsubstitute.dyl
            __TEXT at 0x1021D0000->0x1021EBFFF with permissions r.x
      __DATA_CONST at 0x1021EC000->0x1021EFFFF with permissions rw.
            __DATA at 0x1021F0000->0x1021F3FFF with permissions rw.
        __LINKEDIT at 0x1021F4000->0x1021FA13F with permissions r..
                      0x1021D4000->0x1021E8000 configured for FairPlay decryption
dyld: Mapping /usr/lib/substitute-loader.dylib (slice offset=16384)
dyld: Speculatively read offset=0x00004000, len=0x00325FC0, path=/usr/lib/substitute-loader.dylib
            __TEXT at 0x102A10000->0x102CA3FFF with permissions r.x
      __DATA_CONST at 0x102CA4000->0x102CA7FFF with permissions rw.
            __DATA at 0x102CA8000->0x102CEBFFF with permissions rw.
        __LINKEDIT at 0x102CF0000->0x102D39FBF with permissions r..
                      0x102A14000->0x102CA4000 configured for FairPlay decryption
dyld: Mapping /usr/lib/libsubstrate.dylib (slice offset=16384)
```



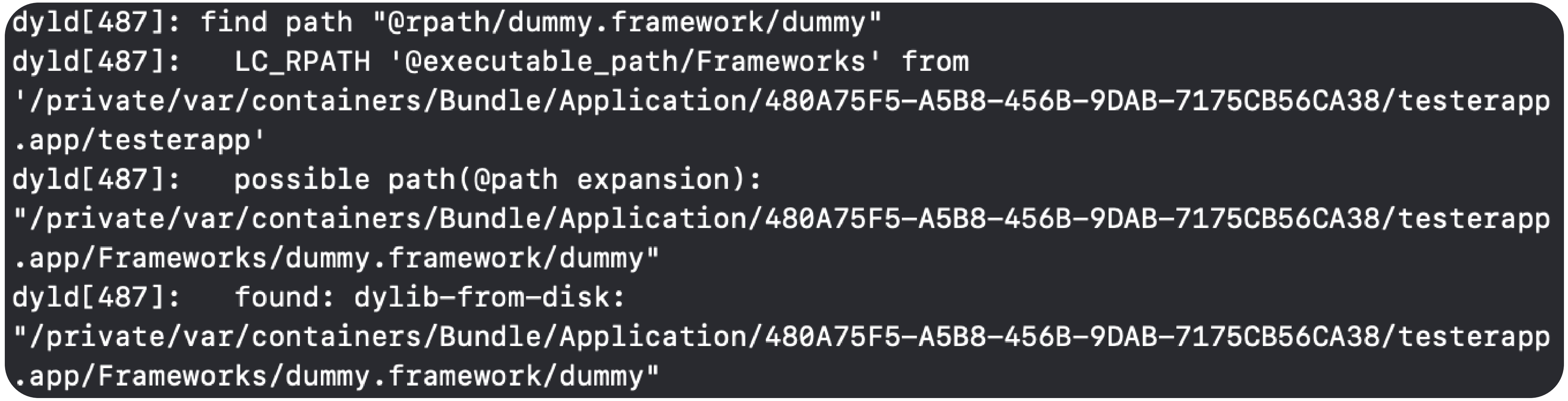
```
dyld: calling -init function 0x19363f2e4 in /System/Library/Frameworks/CoreFoundation.framework/CoreFoundation
dyld: calling initializer function 0x1034e5afc in /usr/lib/substitute-inserter.dylib
dyld: loaded: <181F3AA8-66D9-3165-AC54-344385AC6E1D> /usr/lib/libobjc-trampolines.dylib
dyld: calling initializer function 0x195c87f1c in /usr/lib/libnetwork.dylib
dyld: loaded: <4B291A7E-AE4C-3CE2-82C2-0A3B729E8425> /usr/lib/libsubstitute.dylib
dyld: loaded: <0B9B80FE-7EBD-301D-9FAB-72A262ED54CA> /usr/lib/substitute-loader.dylib
dyld: calling initializer function 0x103720194 in /usr/lib/substitute-loader.dylib
dyld: loaded: <52123F7C-E68A-36DC-8D3F-93DCA9C4DAD6> /usr/lib/libsubstrate.dylib
```

Trace Dylib Hijacking With DYLD_PRINT_SEARCHING



LC_BUILD_VERSION
LC_SOURCE_VERSION
LC_MAIN
LC_ENCRYPTION_INFO_64
LC_LOAD_DYLIB (dummy)
LC_LOAD_DYLIB (Foundation)
LC_LOAD_DYLIB (libobjc.A.dylib)
LC_LOAD_DYLIB (libSystem.B.dylib)
LC_LOAD_DYLIB (CoreFoundation)

Offset	Data	Description	Value
00000A88	0000000C	Command	LC_LOAD_DYLIB
00000A8C	00000038	Command Size	56
00000A90	00000018	Str Offset	24
00000A94	00000002	Time Stamp	Thu Jan 1 02:00:02 1970
00000A98	00010000	Current Version	1.0.0
00000A9C	00010000	Compatibility Version	1.0.0
00000AA0	4072706174682F64756D6D7…	Name	@rpath/dummy.framework/dummy



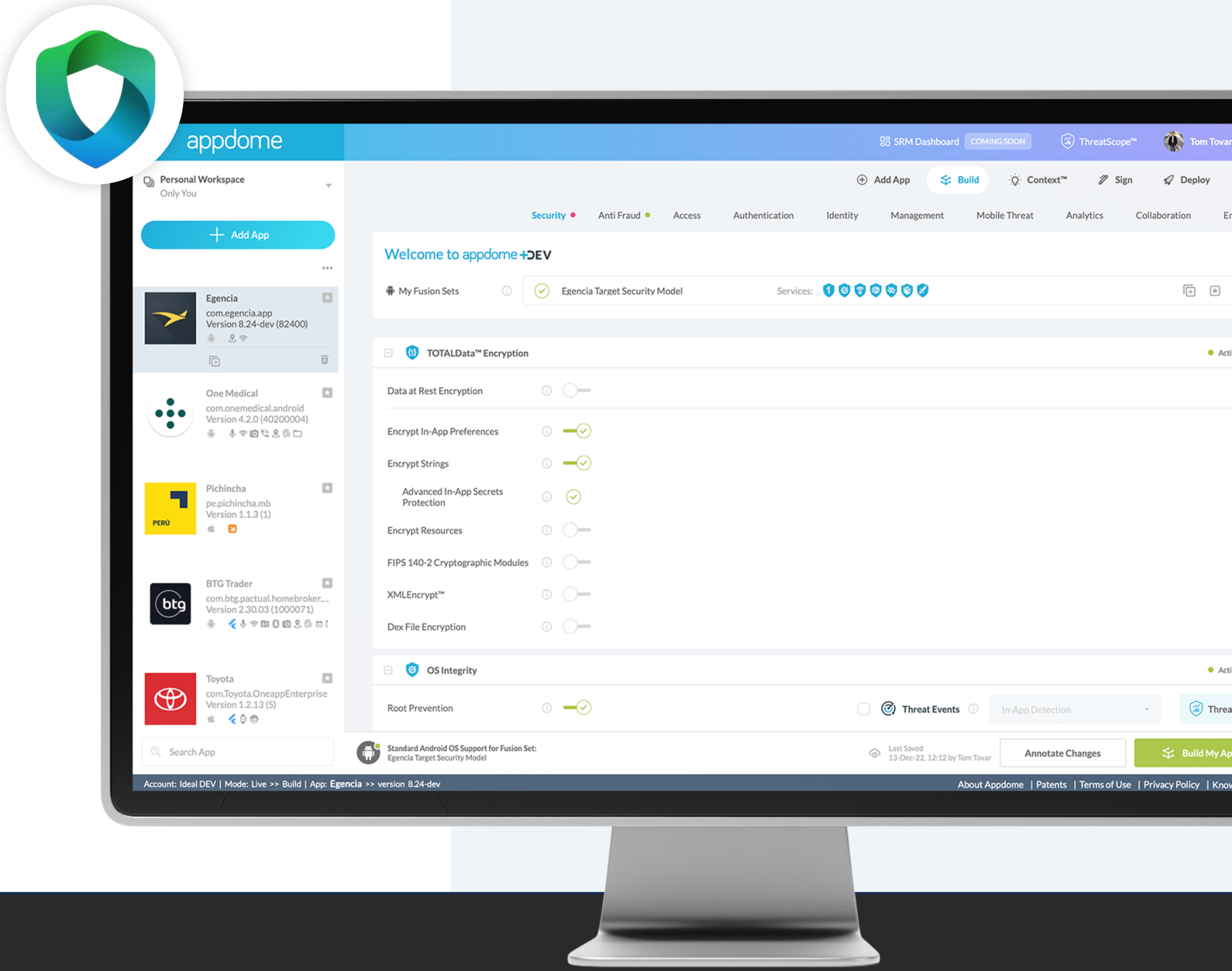
```
dyld[487]: find path "@rpath/dummy.framework/dummy"
dyld[487]:   LC_RPATH '@executable_path/Frameworks' from
'/private/var/containers/Bundle/Application/480A75F5-A5B8-456B-9DAB-7175CB56CA38/testerapp
.app/testerapp'
dyld[487]:   possible path(@path expansion):
"/private/var/containers/Bundle/Application/480A75F5-A5B8-456B-9DAB-7175CB56CA38/testerapp
.app/Frameworks/dummy.framework/dummy"
dyld[487]:   found: dylib-from-disk:
"/private/var/containers/Bundle/Application/480A75F5-A5B8-456B-9DAB-7175CB56CA38/testerapp
.app/Frameworks/dummy.framework/dummy"
```

About Appdome

Our mission is to protect all mobile apps and users from threats and potential risks such as injection techniques, compromised environments, and debugging.

Barak Aharoni

barak@appdome.com | www.appdome.com



Personal Workspace
Only You

Egencia
com.egencia.app
Version 8.24-dev (82400)

One Medical
com.onemedical.android
Version 4.2.0 (40200004)

Pichincha
pe.pichincha.mb
Version 1.1.3 (1)

BTG Trader
com.btg.pactual.homebroker....
Version 2.30.03 (1000071)

Toyota
com.Toyota.OneappEnterprise
Version 1.2.13 (5)

Welcome to appdome +DEV

My Fusion Sets — Egencia Target Security Model — Services:

TOTALData™ Encryption — Acti

Data at Rest Encryption	off
Encrypt In-App Preferences	✓
Encrypt Strings	✓
Advanced In-App Secrets Protection	✓
Encrypt Resources	off
FIPS 140-2 Cryptographic Modules	off
XMLEncrypt™	off
Dex File Encryption	off

OS Integrity — Acti

Root Prevention	✓

Threat Events — In-App Detection — Threa

Standard Android OS Support for Fusion Set:
Egencia Target Security Model

Last Saved
13-Dec-22, 12:12 by Tom Tovar

Account: Ideal DEV | Mode: Live >> Build | App: Egencia >> version 8.24-dev

About Appdome | Patents | Terms of Use | Privacy Policy | Kno

appdome

THANK YOU

Questions?