Ghost in the Droid

Possessing Android Applications with ParaSpectre

Jeff Dileo (chaosdata)

DEFCON 25

Hi!

I’m Jeff, and I have a problem.

I like to do bad things to worse programming languages.

audience says

Hiiiiiiiiiiiiiiiiii Jeff

Outline

- Introduction
- Motivation
- Original Plan
- Android Function Hooking 102
- ParaSpectre
- Demos
- Future Work

Introduction

What is this about?

- Injecting JRuby into Android applications to hook functionality

Why should you care?

- You reverse Android apps
- You develop Android apps, but realize the debugging stack sucks
- You like Ruby and/or REPLs

```
$ irb
irb(main):001:0> puts "this is a REPL"
this is a REPL
=> nil
irb(main):002:0>
$ python
Python 2.7.11 (default, Mar  1 2016, 18:47:52)
[GCC 4.2.1 Compatible Apple LLVM 6.1.0 (clang-602.0.53)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print "this is also a REPL"
this is also a REPL
>>>
```

Motivation

- Was reversing multiple complex Android apps
  - Including a screwy Korean chat app used primarily by Japanese people
- Writing hooks for it was tedious and it was tricky to figure out what all of the nested obfuscated objects were

Original Plan

REPL-ize

- Take the interesting functions
  ...and wrap them in REPLs!

- REPLs are great
  - They give you an interactive shell
  - And let you poke around at stuff

Android Function Hooking — LD_PRELOAD

Shim to Win

`LD_PRELOAD`:

- Old-school function hooking
- `setprop wrap.<pkg> LD_PRELOAD=/path/to/file.so`
- Override dynamically linked native functions
- Inject a native function to run early in app startup
- Requires root access

Android Function Hooking — LD_PRELOAD

Example

```c
#include <dlfcn.h>
#include <stdio.h>
#include <unistd.h>

static int (*_real_rand)(void) = NULL;

__attribute__((constructor))
static void setup() {
  _real_rand = (int(*)(void))dlsym(RTLD_NEXT, "rand");
}

int rand() {
  if(access(".ps3mode", F_OK) != -1 ) {
    return 4;
  }
  return (*_real_rand)();
}
```

Android Function Hooking — Frida

J-J-J-JavaScript!

Frida:

- Stomps over instruction memory to add hooks
- Function hooks (for native code and Java) implemented in JavaScript (or native code using `frida-gum`)
- Injected with either a root daemon, `LD_PRELOAD`, or by modifying an APK
- Requires root access (if not modifying an APK)

Android Function Hooking — Frida

Example

```
Java.perform(function() {
  var File = Java.use('java.io.File');
  File.exists.implementation = function() {
    if(this.path.value == '/system/xbin/su') {
      return false;
    }
    return this.exists();
  }
});
```

Android Function Hooking — Xposed

Monkey-patching zygote is safe, right?

Xposed Framework

- Modifies Zygote to allow for hook code from other packages to be loaded early in the boot of a target application
- Provides an API to register further hooks within an application
- Due to hook code and target application code having different classloaders, hooks generally require a lot of reflection to manipulate instances of classes defined in the target application
- Write hooks in anything that compiles into Java/Dalvik bytecode
- Requires the ability to modify the system image

Android Function Hooking — Xposed

Example (top-level scaffolding)

```
public class XposedEntry implements IXposedHookLoadPackage {
  @Override
  public void handleLoadPackage(XC_LoadPackage.LoadPackageParam lpp)
    throws Throwable {
    if (!lpp.packageName.equals("...")) {
      return;
    }
    ClassLoader singledexcl = lpp.classLoader;
    try {
      <next slide>
    } catch (Throwable t) {...}
  }
}
```

Android Function Hooking — Xposed

Example (multidex scaffolding)

```
XposedHelpers.findAndHookMethod("android.app.Application",
  singledexcl, "attach", Context.class, new XC_MethodHook() {
    @Override
    protected void afterHookedMethod(
      XC_MethodHook.MethodHookParam param) throws Throwable {
      Context context = (Context) param.args[0];
      ClassLoader multidexcl = context.getClassLoader();
      try {
        <next slide>
      } catch (NoSuchMethodError nsme) {
        //pass
      } catch (Throwable t) {...}
    }
  }
);
```

Android Function Hooking — Xposed

Example (main hook)

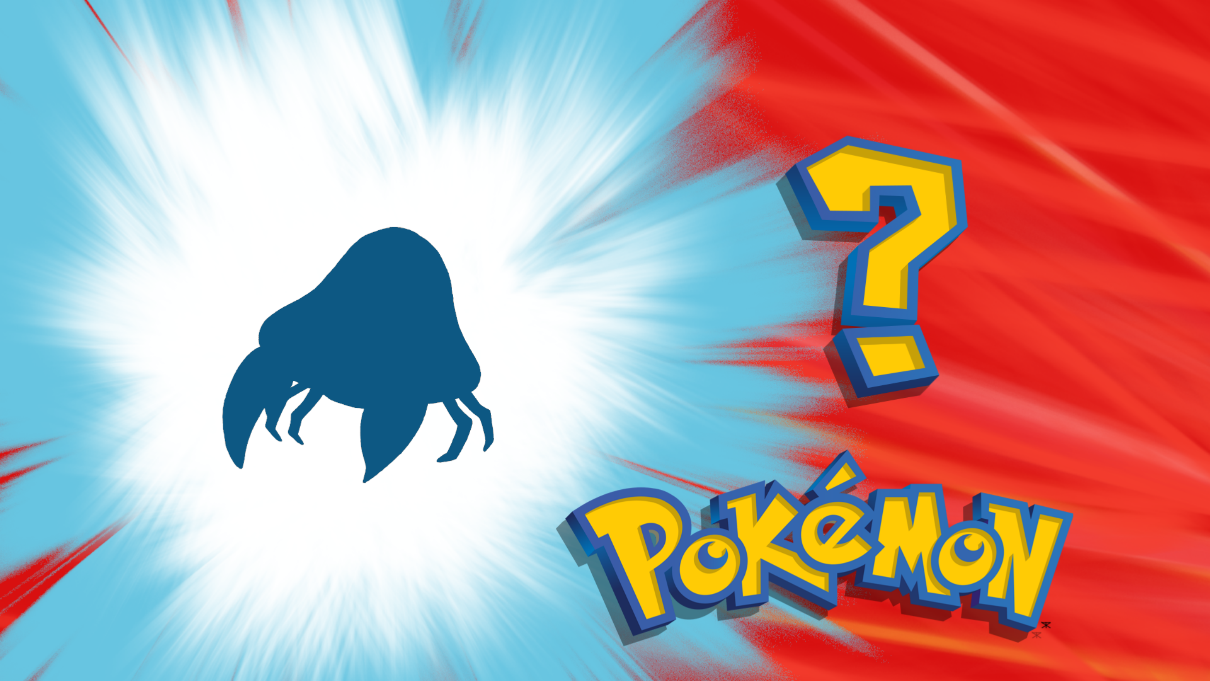
```
XposedHelpers.findAndHookMethod("...", multidexcl, "...",
  ...<...>.class, new XC_MethodHook() {
    @Override
    protected void beforeHookedMethod(
      MethodHookParam param) throws Throwable {
      super.beforeHookedMethod(param);
      ...
    }

    @Override
    protected void afterHookedMethod(
      MethodHookParam param) throws Throwable {
      super.afterHookedMethod(param);
      ...
    }
  }
);
```

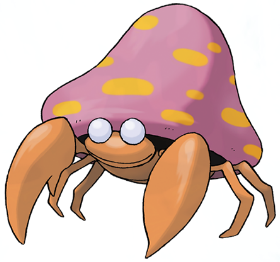
Android Function Hooking — Xposed

Example (actual hook)

```
XposedHelpers.findAndHookMethod(File.class, multidexcl, "exists", new XC_MethodHook() {
  @Override
  protected void beforeHookedMethod(MethodHookParam param) throws Throwable {
    String path = ((File) param.thisObject).getAbsolutePath();
    if (path.equals("/system/xbin/su")) {
      param.setResult(new Boolean(false));
    }
  }
});
```

Note: Bootstrap/Android framework classes don’t require multidex scaffolding to hook.


?
Pokémon



?

Pokémon

Parasect

The ”Mushroom Pokémon”

Pokédex entries:

- Red/Blue
  - *A host-parasite pair in which the parasite mushroom has taken over the host bug. Prefers damp places.*
- Yellow
  - *The bug host is drained of energy by the mushrooms on its back. They appear to do all the thinking.*
- Gold/Stadium 2
  - *It stays mostly in dark, damp places, the preference not of the bug, but of the big mushrooms on its back.*
- Crystal
  - *When nothing’s left to extract from the bug, the mushrooms on its back leave spores on the bug’s egg.*
- Diamond/Platinum/Black(2)/White(2)/X
  - *A mushroom grown larger than the host’s body controls Parasect. It scatters poisonous spores.*

ParaSpectre

"There are only two hard things in Computer Science: cache invalidation and naming things." -Phil Karlton[1]

- **para-**, from Ancient Greek παρά (*pará*, "beside; next to, near, from; against, contrary to")
- in(tro)**spect**ion, from Middle French, from Old French *inspeccion*, from Latin *inspectiō* ("examination, inspection"), from the verb *inspectō* ("I inspect"), from *spectō* ("I observe, I watch"), frequentive of *speciō* ("I look at")
- **spectre**, from French *spectre*, from Latin *spectrum* ("appearance, apparition")
- **Parasect**, from parasite and insect
- **ParaSpectre**, from all of the above

[1] He was an original X11 designer/implementer, so you know he's seen some shit.

ParaSpectre

OK, but seriously, what is it?

- A function/method hooking tool for Android
- Injects a JRuby interpreter into a target process
    - Uses JSON to configure method matching selectors
    - Hooked functions call into custom Ruby code
        - And/or drop into an interactive in-process Ruby REPL
- Implemented using Xposed
    - Provides first class access to the Java runtime environment and classloaders
    - Ensures that arbitrary app packages may be hooked at device startup
- Hook reloading only requires restarting the application/process
    - For reference, reloading Xposed hooks themselves requires reinstalling the hook app's APK and then rebooting the device.

Capabilities

Let your hooks choose their own destiny!

Matching selectors

- Be as specific or vague as you want to select methods for hooking
    - Uses an intersection of the provided selectors to filter
- Class matching (if class name is not supplied), by:
    - superclass name
    - implemented interfaces
- Method matching, by:
    - method name
    - argument type signature
    - return type
    - exception signature

Capabilities

MINASWAN

Ruby (via JRuby)

- Solid scripting language
  - Can be forced to run on Android
    - ...with relatively minimal blood sacrifices
- Solid Java interop made better with classloader injection
  - Code runs with access to the hooked application’s classloader
    - No need for reflection, just write the code
    - Define subclasses/impls for app-defined classes/ifaces and plug them
- Stackable script hooks
  - Per application package
  - Per class matcher
  - Per method matcher

When this card is Special Summoned: You can Special Summon as many "Crystal Beast" cards as possible from your Spell & Trap Card Zone. If this face-up card is destroyed while it is in a Monster Card Zone, you can place it face-up in your Spell & Trap Card Zone as a Continuous Spell Card, instead of sending it to the Graveyard.

Capabilities

Run wild at runtime!

Runtime exploration

- With Pry[2] REPLS!
    - Pry is a suped-up REPL for Ruby, it’s way better than IRB
- Drop to a Pry REPL to inspect and manipulate application state at runtime
- By default, hooks will drop into a Pry REPL if they don’t `return` early

[2] `https://pryrepl.org`

Features

Connect-back REPLs

- Uses a modified version of `pry-remote`[3]
  - Modifies how it uses the DRuby distributed object protocol
    - Adds support for specifying client and daemon ports
    - Adds support for Unix domain sockets
    - Add authentication (see below)
- Uses a modified Ruby stdlib and a custom authenticating proxy that adds authentication to DRuby
  - If you couldn't tell by now, DRuby is a super dangerous protocol that is completely unauthenticated and, by default, enables RCE
- Each connect-back REPL is opened in a new `tmux` window
- Injects hooks into the package manager system service to enable the main ParaSpectre app to grant the `INTERNET` permission to apps that don't request it.

[3] `https://github.com/chaosdata/pry-remote`

Features

You did WHAT with Jetty?!?

Includes a configuration editor web application

- Raw Jetty Servlet[4] web app running on Android
    - Usable from a mobile browser on the Android device itself!
- Used to configure method matcher selectors and write Ruby hook code
- Supports a hook editing workflow that doesn't require `adb push`
- UI is Ace-based[5]
- Edits are tracked in an on-device Git repo
- Basic access controls using API keys regenerated on web app start
- Per-app hook config files, with format validation
- Write inline Ruby hooks or reference flat Ruby files

[4] Undertow and RESTEasy had issues due to AWT dependencies

[5] `https://ace.c9.io`

Design

”Simple” in the sense that this fits on a slide

- Loads hook configuration data
  - Reads (`rw-r-r--`) config files from main ParaSpectre app directory
    - Based on app package name
    - Falls back to a core `paraspectre.json` config
- Sets up a JRuby environment on Android
  - Xposed hook loads pre-dexed JRuby JAR into a hook-configured application
  - Uses some reflection-based environment setup, options tweaking, and custom classes added into JRuby to make it run properly on Android
- Iterates through all classes in target application’s classloader chain
- Selectors use config values to pick from available classes
- Uses Xposed to set up hooks on matching classes/methods
- The Xposed hooks invoke the config-specified JRuby

Hooks

Instant ramen hook

The JSON config format is a work in progress, but works well enough.

```
{
  "classes": [
    {
      "name": "android.support.v7.app.AppCompatActivity",
      "methods": [
        {
          "name": "findViewById",
          "params": ["int"],
          "returns": "android.view.View",
          "eval": "puts 'id: ' + args[0].to_s; return;"
        }
      ],
      "eval": "puts 'in ' + method.to_s;"
    }
  ],
  "eval": ""
}
```

Hooks — Configuration

”Jay Sahn”

More involved hooks should be broken out into a separate Ruby file.

```
{
  "classes": [
    {
      "name": "okhttp3.OkHttpClient$Builder",
      "methods": [
        {
          "name": "build",
          "eval_file": "okhttp3.OkHttpClient$Builder::build.rb"
        }
      ]
    }
  ]
}
```

Hooks — Code

"Jay Roo Bee"

```
this.proxy(java.net.Proxy.new(
  java.net.Proxy::Type.valueOf('HTTP'),
  java.net.InetSocketAddress.new('127.0.0.1',8080))
)
this.certificatePinner(
  Java::Okhttp3.CertificatePinner::DEFAULT
);

trustAllCerts = Class.new() {
  include javax.net.ssl.X509TrustManager
  def checkClientTrusted(chain,authType)
  end
  def checkServerTrusted(chain,authType)
  end
  def getAcceptedIssuers()
    [].to_java(java.security.cert.X509Certificate)
  end
}.new
```

```
ctx = javax.net.ssl.SSLContext.getInstance('SSL')
ctx.init(
  nil, [trustAllCerts],
  java.security.SecureRandom.new
)
socketFactory = ctx.getSocketFactory()

this.sslSocketFactory(socketFactory, trustAllCerts)
verifier = Class.new() {
  include javax.net.ssl.HostnameVerifier
  def verify(hostname,session)
    true
  end
}.new
this.hostnameVerifier(verifier)


return
```

Performance Tricks

JRuby Initialization

- Pre-dexed JRuby jar is loaded into the classloader during Zygote init
  - Due to SEAndroid policies, stores this file under `/data/dalvik-cache/paraspectre`
    - Zygote can read from it, runtime root can write to it
- Due to race conditions inherent in Android's boot sequence, attempting to initialize a JRuby script container in Zygote deadlocks the system due to Zygote taking too long to initialize
  - Dianne Hackborn, please save us from this darkness[6]
- As a result, JRuby scripting containers are initialized separately in each hooked app
  - This is time consuming
  - But we can kick this off in a background thread at the Xposed entry point in app start
- The initial run of Ruby code in an initialized container takes several seconds to run
  - Post-init, a Ruby hook script of "return;" is `eval`'d in the container to prep it before use

[6] Also, can you kill D-Bus and replace it with binder?

Performance Tricks

Class searching and matching

- Various performance tricks played in scanning classes for matchers
  - To search, it needs to iterate through the list of loaded classes
    - Save time here by only iterating through class names in app's own DEX files
  - Normal `ClassLoader::loadClass` hits a worst-case path where it searches through the parent classloader for framework classes
    - Bypassed this by yanking out the `protected` `dalvik.system.BaseDexClassLoader::findClass` method and invoking it directly
  - Still running into the classloader global lock
    - This prevents multithreaded class iteration, and actually makes it less performant due to lock contention
    - May eventually parse DEX files directly to get metadata for matchers

Performance

Results

- JRuby container initialization went from 29 seconds of startup overhead to being nigh-instantaneous*
- Class matching overhead is generally unobservable on single DEX applications
    - `com.facebook.katana`[7] has 12 `classes.dex` files comprising about 100k classes; it is not a slender blade
        - Class iteration (not performed if class matchers are specified by name) takes 30 seconds
        - Once iterated, the matching set of classes (logged to `logcat`) can be specified by name in the config

[7] Literally the biggest Android app I can think of.

Performance

Caveats

If a hook runs automatically on startup, it may have to wait for the initial JRuby container to be fully initialized, which can take up to 6 seconds on a ”modern” Android device[8]

- This runs in parallel to any class searching, which fully blocks app startup to prevent target methods from running unhooked

[8] All Android performance numbers come from a Nexus 5X.

Performance

Speed and Latency

- Overall though, the edit workflow is two orders of magnitude smaller than writing raw Xposed hooks
    - Edit Java code (??)
    - Compile Java code as an Android app (30s+)[9]
    - Copy APK to mobile device (10s+)
    - Install APK (30s+)
    - Reboot phone (2-3 minutes if the device is encrypted and has a PIN)

[9] All laptop performance numbers come from a Late 2013 13” MacBook Pro.

Demos

Where?

Soon

https://github.com/nccgroup/paraspectre

Current Limitations

Caveat emptor!

- The DRuby protocol is scary, a hooked app (as it can authenticate) can potentially gain RCE on the host running the `pry-remote`-based client
    - For now, it’s probably best to run the REPL client from a VM
    - Long term solution involves research into DRuby
    - Medium term solution involves sandboxing the client
- Adding gems is not supported yet, and requires manual bit twiddling

Future Work

Fixing the limitations

- Gem JAR file upload API
- Overhaul the UI for creating, editing, and managing hooks
- Android 7/N+ compatibility (once Xposed supports it)
    - Current world-readable config file implementation may break due to SEAndroid changes
    - Google’s workaround uses the Android support library, not a standard class
    - Leveraging root access to edit a shared config in the `/data/dalvik-cache/paraspectre` directory is ugly, but feasible
- Figure out the DRuby situation

Greetz

Here’s to all the little people…

- aleks
- arkos
- bones
- justin
- nabla
- niko
- weber

Questions?

jeff.dileo@nccgroup.trust

@ChaosDatumz

Ghost in the Droid

Possessing Android Applications with ParaSpectre

Jeff Dileo (chaosdata)

DEFCON 25