

# Stealth File Systems for Proactive Forensics Support in Custom Android ROMs

Guide: Dr. Prabhaker Mateti<sup>1</sup>    Sudip Hazra<sup>2</sup>

<sup>1</sup>Wright State University

<sup>2</sup>Amrita Centre For Cyber Security Systems and Networks  
Amrita University

13 March 2017



## 1 Introduction

- Why Android
- Why Mobile Forensics is Important
- Problem Scenario

## 2 Background

- Types of Mobile Forensics Investigation

- Forensic Support Framework
- Shortfalls
- Possible Solutions
- File System in User Space
- Cloud File System

## 3 Problem Definition

## 4 Solution Outline

- Modules
- Cloud Storage API's

- Calling Native Libraries Using Java
- Calling Native Code using JNA
- Addressing Performance Issues
- Stealth

## 5 Implementation

## 6 Summary

## 1 Introduction

- Why Android
- Why Mobile Forensics is Important
- Problem Scenario

## 2 Background

- Types of Mobile Forensics Investigation

- Forensic Support Framework
- Shortfalls
- Possible Solutions
- File System in User Space
- Cloud File System

## 3 Problem Definition

## 4 Solution Outline

- Modules
- Cloud Storage API's

- Calling Native Libraries Using Java
- Calling Native Code using JNA
- Addressing Performance Issues
- Stealth

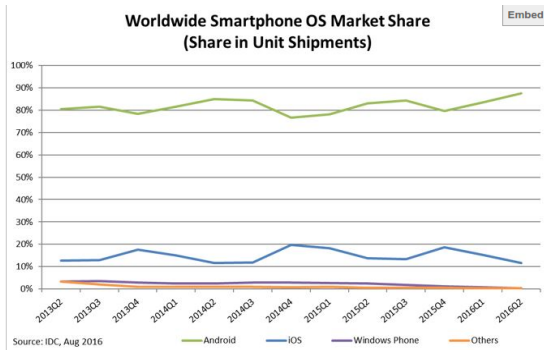
## 5 Implementation

## 6 Summary

# Why Android

## Mobile OS Market

Figure : Smartphone OS Market Share, 2016 Q2



Source: <http://www.idc.com/prodserv/smartphone-os-market-share.jsp>

## 1 Introduction

- Why Android
- Why Mobile Forensics is Important
- Problem Scenario

## 2 Background

- Types of Mobile Forensics Investigation

- Forensic Support Framework
- Shortfalls
- Possible Solutions
- File System in User Space
- Cloud File System

## 3 Problem Definition

## 4 Solution Outline

- Modules
- Cloud Storage API's

- Calling Native Libraries Using Java
- Calling Native Code using JNA
- Addressing Performance Issues
- Stealth

## 5 Implementation

## 6 Summary

# Why Mobile Forensics is Important

The use of cell phones and computers are like a journal or diary of users lives. Just from cell phones, a mobile phone forensic analysis by International Investigators can reveal a great deal of data, including:

- Dialed, incoming and missed calls (history logs)
- Text messages
- Instant message activity
- Email
- Internet activity including search histories
- Phone location information (using GPS) and cell phone tower triangulation

## 1 Introduction

- Why Android
- Why Mobile Forensics is Important
- Problem Scenario

## 2 Background

- Types of Mobile Forensics Investigation

- Forensic Support Framework
- Shortfalls
- Possible Solutions
- File System in User Space
- Cloud File System

## 3 Problem Definition

## 4 Solution Outline

- Modules
- Cloud Storage API's

- Calling Native Libraries Using Java
- Calling Native Code using JNA
- Addressing Performance Issues
- Stealth

## 5 Implementation

## 6 Summary

## To Catch a Thief, Think Like a Thief!

- If criminals and crime organizers use smart phones, what would they do?
- Will they browse? If so which browser? What site?
- How to predict the next move?
- How to Collect Evidence if they erase the Phone Memory aka. Factory Reset.



## 1 Introduction

- Why Android
- Why Mobile Forensics is Important
- Problem Scenario

## 2 Background

- Types of Mobile Forensics Investigation

- Forensic Support Framework
- Shortfalls
- Possible Solutions
- File System in User Space
- Cloud File System

## 3 Problem Definition

## 4 Solution Outline

- Modules
- Cloud Storage API's

- Calling Native Libraries Using Java
- Calling Native Code using JNA
- Addressing Performance Issues
- Stealth

## 5 Implementation

## 6 Summary

# Types of Mobile Forensics Investigation

- **Reactive Forensics :**

Investigation done after Crime has happened. Susceptible to Applications like Uninstall-It, can potentially wipe out all user data and Device Encryption can be a barrier for investigation.

- **Proactive Forensics:**

A suspect or potential terrorist is monitored proactively in realtime to prevent a crime. Real-time monitoring and analysis is possible. Data Encryption will not be a hindrance in evidence collection. Ability to retrieve deleted data and logs and prevent potential crimes .

## 1 Introduction

- Why Android
- Why Mobile Forensics is Important
- Problem Scenario

## 2 Background

- Types of Mobile Forensics Investigation

## • Forensic Support Framework

- Shortfalls
- Possible Solutions
- File System in User Space
- Cloud File System

## 3 Problem Definition

## 4 Solution Outline

- Modules
- Cloud Storage API's

- Calling Native Libraries Using Java

- Calling Native Code using JNA
- Addressing Performance Issues
- Stealth

## 5 Implementation

## 6 Summary

# Forensic Support Framework

**Figure :** Features of forensic Rom developed by Aiyappan Et.al [1] and Karthik Et.al [2]

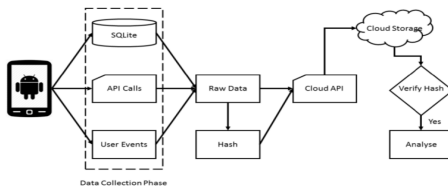


Figure: Forensic Service Architecture

- 1 .Captures All User Activities.
- 2 .Key-logging and Call Tapping Facility.
- 3 .Opportunistically Uploads In Cloud.
4. Hiding the Process using hidepid =2.
5. Data Stored in /forensic partition only accessible to Root.

## 1 Introduction

- Why Android
- Why Mobile Forensics is Important
- Problem Scenario

## 2 Background

- Types of Mobile Forensics Investigation

- Forensic Support Framework

### • Shortfalls

- Possible Solutions
- File System in User Space
- Cloud File System

## 3 Problem Definition

## 4 Solution Outline

- Modules
- Cloud Storage API's

- Calling Native Libraries Using Java

- Calling Native Code using JNA
- Addressing Performance Issues
- Stealth

## 5 Implementation

## 6 Summary

# Shortfalls

- What if The Suspect Roots the Phone ?
- Can Find the /forensic Partition.

## 1 Introduction

- Why Android
- Why Mobile Forensics is Important
- Problem Scenario

## 2 Background

- Types of Mobile Forensics Investigation

- Forensic Support Framework

- Shortfalls

## • Possible Solutions

- File System in User Space
- Cloud File System

## 3 Problem Definition

## 4 Solution Outline

- Modules
- Cloud Storage API's

- Calling Native Libraries Using Java

- Calling Native Code using JNA

- Addressing Performance Issues

- Stealth

## 5 Implementation

## 6 Summary

- Encrypting The /forensic partition can Still arise Suspicion.
- Creating A Fuse File System and enable Stealth Features and Copy all Forensically Relevant Data in that File System.



## 1 Introduction

- Why Android
- Why Mobile Forensics is Important
- Problem Scenario

## 2 Background

- Types of Mobile Forensics Investigation

- Forensic Support Framework
- Shortfalls
- Possible Solutions
- File System in User Space
- Cloud File System

## 3 Problem Definition

## 4 Solution Outline

- Modules
- Cloud Storage API's

- Calling Native Libraries Using Java
- Calling Native Code using JNA
- Addressing Performance Issues
- Stealth

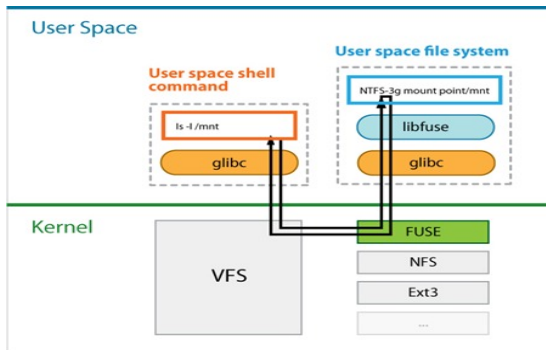
## 5 Implementation

## 6 Summary

# File System in User Space

- The Filesystem in Userspace (FUSE) is a special part of the Linux kernel that allows regular users to make and use their own file-systems without needing to change the kernel or have Root privileges.

Figure : A Fuse Filesystem.



Source: [en.wikipedia.org/wiki/Filesystem\\_in\\_Userspace](https://en.wikipedia.org/wiki/Filesystem_in_Userspace)

## 1 Introduction

- Why Android
- Why Mobile Forensics is Important
- Problem Scenario

## 2 Background

- Types of Mobile Forensics Investigation

- Forensic Support Framework
- Shortfalls
- Possible Solutions
- File System in User Space
- Cloud File System

## 3 Problem Definition

## 4 Solution Outline

- Modules
- Cloud Storage API's

- Calling Native Libraries Using Java
- Calling Native Code using JNA
- Addressing Performance Issues
- Stealth

## 5 Implementation

## 6 Summary

# Cloud File System

- Using FUSE we can mount Cloud Drive in Our System and Use it Like a Local File System.
- Gcsfuse: A user-space file system for interacting with Google Cloud Storage.
- Wingfs: A debian Package to mount various cloud storage drives as user-space file systems.
- Azurefs: A python package to mount Azure blob storage as Local File system.

# Problem Definition

## What is the Goal

- To mount the Cloud storage as a Local File system in Android.
- To Provide Support for Multiple Cloud storage Providers.
- The forensic file system will copy itself in parts to the cloud file system.
- The file system will be opportunistically get uploaded to the cloud storage.
- To hide both the Cloudfs and the forensic partition using Rootkits.

# Architecture Diagram

Here we propose a Stealth File system with cloud support Below the Android Software stack.

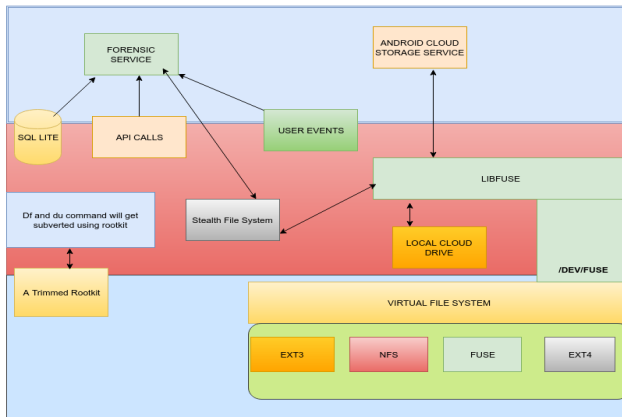


Figure : Android Cloud Storage Service

## 1 Introduction

- Why Android
- Why Mobile Forensics is Important
- Problem Scenario

## 2 Background

- Types of Mobile Forensics Investigation

- Forensic Support Framework
- Shortfalls
- Possible Solutions
- File System in User Space
- Cloud File System

## 3 Problem Definition

## 4 Solution Outline

- Modules
- Cloud Storage API's

- Calling Native Libraries Using Java
- Calling Native Code using JNA
- Addressing Performance Issues
- Stealth

## 5 Implementation

## 6 Summary

- The stealth file system will be a separate file system which will be used by the forensic service to copy all the forensically relevant data from the relevant device partitions like /data to the stealth file system.
- The stealth file system will be based on ext4 file systems, rootkits will be used to hide the file volumes from commands like df and du.
- The rootkit will also hide the forensic service running in the background by hooking on to the sys\_call\_table and filtering out the output.



# Modules

## Cloud File System

- The cloud File system will be made using fuse , and we will be able to locally mount a cloud drive as the cloud file system. It will support various cloud providers using apis
- **As per the official android documentation the external storage (SD cards) are accessed by the Android system using FUSE which implies that FUSE is supported by the kernel directly so we don't need to add fuse support in kernel.**

## 1 Introduction

- Why Android
- Why Mobile Forensics is Important
- Problem Scenario

## 2 Background

- Types of Mobile Forensics Investigation

- Forensic Support Framework
- Shortfalls
- Possible Solutions
- File System in User Space
- Cloud File System

## 3 Problem Definition

## 4 Solution Outline

- Modules
- Cloud Storage API's

- Calling Native Libraries Using Java
- Calling Native Code using JNA
- Addressing Performance Issues
- Stealth

## 5 Implementation

## 6 Summary

## Cloud Storage API's Examples:

Dropbox Cloud Storage Api's:

1 .Create a Dropbox folder

```
post('https://api.dropbox.com/1/fileops/create_folder', args)
```

2.Rename a Dropbox file/directory object.

```
post('https://api.dropbox.com/1/fileops/move', args)
```

3.Delete a Dropbox file/directory object.

```
post('https://api.dropbox.com/1/fileops/delete', args)
```

4.Get Dropbox metadata of path.

```
get('https://api.dropbox.com/1/metadata/auto' + path, args)
```

## 1 Introduction

- Why Android
- Why Mobile Forensics is Important
- Problem Scenario

## 2 Background

- Types of Mobile Forensics Investigation

- Forensic Support Framework
- Shortfalls
- Possible Solutions
- File System in User Space
- Cloud File System

## 3 Problem Definition

## 4 Solution Outline

- Modules
- Cloud Storage API's

## • Calling Native Libraries Using Java

- Calling Native Code using JNA
- Addressing Performance Issues
- Stealth

## 5 Implementation

## 6 Summary

# Calling Native Libraries Using Java

Since libfuse is a native library written in c, We need to have some wrapper functions to call the Native Libraries in Java.

This can be achieved by :

1. Java Native Interface(JNI).
2. Java Native Access(JNA).
3. Java Native Runtime(JNR).

## 1 Introduction

- Why Android
- Why Mobile Forensics is Important
- Problem Scenario

## 2 Background

- Types of Mobile Forensics Investigation

- Forensic Support Framework
- Shortfalls
- Possible Solutions
- File System in User Space
- Cloud File System

## 3 Problem Definition

## 4 Solution Outline

- Modules
- Cloud Storage API's

- Calling Native Libraries Using Java

## • Calling Native Code using JNA

- Addressing Performance Issues
- Stealth

## 5 Implementation

## 6 Summary

# Calling Native Code using JNA

```
Package net.fusejna.examples;  
import java.io.File;  
import java.nio.ByteBuffer;  
import net.fusejna.DirectoryFiller;  
import net.fusejna.ErrorCodes;  
import net.fusejna.FuseException;  
import net.fusejna.StructFuseFileInfo.FileInfoWrapper;  
import net.fusejna.StructStat.StatWrapper;  
import net.fusejna.types.TypeMode.NodeType;  
import net.fusejna.util.FuseFilesystemAdapterFull;
```

We need to Subclass net.fusejna.FuseFilesystem and override the methods we need i.e FuseFilesystemAdapterFull.

```
public int readdir(final String path, final DirectoryFiller filler)  
filler.add(filename); return 0;
```

## 1 Introduction

- Why Android
- Why Mobile Forensics is Important
- Problem Scenario

## 2 Background

- Types of Mobile Forensics Investigation

- Forensic Support Framework
- Shortfalls
- Possible Solutions
- File System in User Space
- Cloud File System

## 3 Problem Definition

## 4 Solution Outline

- Modules
- Cloud Storage API's

- Calling Native Libraries Using Java
- Calling Native Code using JNA
- Addressing Performance Issues
- Stealth

## 5 Implementation

## 6 Summary



# Addressing Performance Issues

Android imposes user-level filesystem (FUSE) over native filesystem partition to provide flexibility in managing the internal storage space and to maintain host compatibility. However, the overhead of user-level filesystem is prohibitively large and the native storage bandwidth is significantly under-utilized. In order to address this overhead of user-level filesystem and Compensate the JNA Overhead. We will be using Buffered Fuse[4]

The key technical ingredients of Buffered FUSE[4] are

- (i) extended FUSE IO size.
- (ii) internal user-level write buffer (FUSE buffer).
- (iii) independent management thread which performs time driven FUSE buffer synchronization.

In this paper[4] the author has increased the fuse buffer size to 512 kb and FUSE Buffer , also the android ioschededular has been changed and file system has been changed to XFS. They are claiming 70 % increase in speed.The modified sdcard.c has the following changes on study:

In android/platform\_system\_core/sdcard.c:

They have defined a FUSE CACHE.

```
#include pthread.h
```

```
#define FUSE_CACHE 1
```

```
#define FUSE_CACHE_THREAD 1
```

```
#define FUSE_CACHE_SIGNAL 0
```

```
#define FUSE_IO_SIZE 512*1024
```

```
#define FUSE_CACHE_SIZE FUSE_IO_SIZE*4
```

```
#define FUSE_CACHE_NUM 4
```

## 1 Introduction

- Why Android
- Why Mobile Forensics is Important
- Problem Scenario

## 2 Background

- Types of Mobile Forensics Investigation

- Forensic Support Framework
- Shortfalls
- Possible Solutions
- File System in User Space
- Cloud File System

## 3 Problem Definition

## 4 Solution Outline

- Modules
- Cloud Storage API's

- Calling Native Libraries Using Java
- Calling Native Code using JNA
- Addressing Performance Issues

## • Stealth

## 5 Implementation

## 6 Summary

- The vector table is a table that actually contains control transfer instructions that jump to the respective exception handlers.
- On 32-bit ARM Linux this address is 0xffff0000.
- Our plan is going to be to backdoor the kernel by overwriting the SWI exception vector with code that jumps to our backdoor code. This code will check for a magic value in a register and if it matches, it will elevate the privileges of the calling process.
- If we choose an appropriate location in kernel space, our code will exist as long as the machine is running

# Implementation

## Compiled Android kernel with module support

- Android kernel, by default do not support loadable modules.
- `make ARCH=arm goldfish_defconfig` (This will create our `.config`)
- Edit the `.config` file and searched for the line `CONFIG_MODULES` It was unset for me and the line looked liked this
- `CONFIG_MODULES` is not set I changed the line to `CONFIG_MODULES=y`, saved and closed the file.
- Now, the command to build the kernel `make ARCH=arm CROSS_COMPILE= /Desktop/mydroid/prebuilt/linux-x86/toolchain/arm-eabi-4.4.3/bin/arm-eabi-`

# Summary






- **Process Hiding** can also be Implemented Using this Technique
- The **Stealth File-system** will periodically copy the forensically relevant data from the normal file system
- This data will be moved to the **Mounted Cloud Drive** and opportunistically uploaded to the cloud server.

# Summary

## Summary

This Framework can effectively Hide the forensic as well as the cloud file system so that even if the Suspect is connecting to adb to check the internal state , He will not be able to find the hidden File systems.

# References I

-  *Android forensic support framework*, Aiyappan.P Advisor:Prabhaker Mateti, M.Tech thesis, Amrita Vishwa Vidyapeetham,2015
-  *Proactive Forensic Support for Android Devices*, Karthik K. Advisor:Prabhaker Mateti M.Tech thesis, Amrita Vishwa Vidyapeetham,2016
-  *Android platform based linux kernel rootkit*, Dong-Hoon You, Bong-Nam Noh, Malicious and Unwanted Software (MALWARE), 2011 6th International Conference ,IEEE
-  *Buffered FUSE: optimising the Android IO stack for user-level filesystem*, Jeong, Sooman and Won, Youjip, International Journal of Embedded Systems,2014.
-  *Corrupting the ARM Exception Vector Table*,  
<http://doar-e.github.io/blog/2014/04/30/corrupting-arm-evt/>