IJCAT International Journal of Computing and Technology, Volume 1, Issue 2, March 2014
ISSN : 2348 - 6090
www.IJCAT.org

Implementing Deletion Recovery on Android

[1]Akhil C Joy, [2]Arjun V, [3] Mithun K, [4] Shebix Xavier

[1, 2, 3, 4] Department of Information Technology, Jyothi Engineering College
Cheruthuruthy,Thrissur, Kerala, India

Abstract - The mobile phones data is deleted once, it cannot be retrieved. In this project we try to develop a new application "Rebin" for avoiding this situation. on the Windows desktop recycle bin represents a directory where deleted files are temporarily stored. This enables one to retrieve files that he may have accidentally deleted. From time to time, it helps to *purge* the recycle bin to free up space on the hard disk. It can also be configured Windows so that it doesn't use the recycle bin at all, but then it won't be able to retrieve accidentally deleted files. By default, Microsoft Windows 95 and above uses 10% of available disk space to save any deleted files in case any file is accidentally deleted it can be recovered from the Recycle Bin. Below is additional information about how to view deleted files, manage the Recycle Bin, and empty the Recycle Bin. The recycle bin is modeled after the Macintosh trash can, which has been part of the Mac GUI since its inception. If any file has been deleted, it is automatically moved to the Recycle Bin. Users can identify if files are in the Recycle Bin by looking at the Recycle Bin icon. By default, the icon will look like the icon to the right, an empty Recycle Bin. When files have been deleted, the Recycle Bin will be full of trash.

***Keywords* - File Recovery, Recycle Bin, Android, Deleted File Recovery, Android Application.**

1. Introduction

In mobile phones data is deleted once, it cannot be retrieved. In this project we try to develop a very new application "Rebin" for avoiding this situation. on the Windows desktop recycle bin represents a directory where deleted files are temporarily stored. This enables one to retrieve files that he may have accidentally deleted. From time to time, It helps to *purge* the recycle bin to free up space on the hard disk. It can also be configured Windows so that it doesn't use the recycle bin at all, but then it won't be able retrieve accidentally deleted files. Similar to the Apple Macintosh Trash, the Recycle Bin is a location where deleted files are temporarily stored on Microsoft Windows 95, 98, ME, NT, 2000, XP, Vista, and all later versions of Microsoft Windows. The Recycling Bin allows users to recover files that have been deleted in Windows. In the image to the right, is an example of what the empty Recycle Bin may look like in your version of Windows and can be found on the Desktop. By default, Microsoft Windows 95 and above uses 10% of available disk space to save any deleted files in case any file is accidentally deleted it can be recovered from the Recycle Bin. Below is additional information about how to view deleted files, manage the Recycle Bin, and empty the Recycle Bin. The recycle bin is modeled after the Macintosh trash can, which has been part of the Mac GUI since its inception. If any file has been deleted, it is automatically moved to the Recycle Bin. Users can identify if files are in the Recycle Bin by looking at the Recycle Bin icon. By default, the icon will look like the icon to the right, an empty Recycle Bin. When files have been deleted, the Recycle Bin will be full of trash.

When we delete a file from in Windows 8/7/Vista/XP PC hard disk, windows operating system will send it to Recycle Bin. But if it is deleted a file from external drive like USB drive, external hard disk, the file will be deleted directly, because Recycle Bin sits on local hard drive, not external drive. Also, Recycle Bin has a limited size. If file size is too large, or Recycle Bin does not have enough space to store the file sent to Recycle Bin, the file will be deleted directly. The good news is that all files deleted (even you deleted them from Recycle Bin) are still in the hard disk.

There will have chance to recover those files if they are not overwritten by new files. The bad news is that when you save new file, This may be stored to the space of deleted files. Once overwritten, deleted files will not be able to be recovered any more. So it's very important to not use computer after files are deleted. At present , mobile phones are using 'Delete' option for deleting data. If the data is deleted once, it permanently disappears from the memory. We have to use Data Recovery Software to get back the data. Here we propose the project which consists the idea of a windows recycling bin. We are supposed to recover and restore data files which are unexpectedly deleted from the mobile system by using an application software "Rebin". This approach will be used for restoring files in a secure mode. The main feature of this project includes restoring the deleted data files. It will

restore the data with a single tap. Requirement of Internet is not needed for this application.

Any item that is generally deleted is present in the recycle bin. One can access the recycle bin and have the option to restore the previously deleted item or alternatively completely delete the recycle bin leading to a permanent deletion of the item previously present in the recycle bin. In the present situation , mobile phones are using 'Delete' option for deleting data. If the data is deleted once, it permanently disappears from the memory. since android devices are not like computers which has a hard disk, a large storage device, android device has very limited internal memory, and even with pluggable SD card or other memory card, the storage space is limited. The answer is Android Data Recovery Software. When android files are deleted, the android system does not erase the actual data on android storage, it only modify the file table entry and delete those files entry. It's fast and serve the purpose well as system can store new files to the space of deleted files. So if deleted files space is overwritten, android data recovery software is able to scan the android memory and look for deleted files data, then data recovery software can restore deleted files with those data. This creates a big in convenience to the mobile users. Therefore we hereby try to implement this project. We are supposed to recover and restore data files which are unexpectedly deleted from the mobile system by using an application software "Rebin". This approach will be used for restoring files in a secure mode. The main feature of this project includes restoring the deleted data files. It will restore the data with a single tap. Requirement of Internet is not needed for this application. This application can only be used in android OS. No one except the administrator cannot restore or permanently delete the files. Android is an operating system for mobile devices such as smart phones and tablet computers. It is developed by the Open Handset Alliance led by Google.

2. Related Work

2.1 Implementing and Optimizing an Encryption File System on Android

The recent surge in popularity of smart hand held devices, including smart-phones and tablets, have given rise to new challenges in protection of Personal Identifiable Information (PII). Indeed, modern mobile devices store PII for applications that span from email to SMS and from social media to location-based services increasing the concerns of the end user's privacy. Therefore, there is a clear need and expectation for PII data to be protected in the case of loss, theft, or capture of the portable device. In this paper, we present a novel FUSE (File system in USErspace) encryption file system to protect the removable and persistent storage on heterogeneous smart gadget devices running the Android platform. The proposed file system leverages NIST certified cryptographic algorithms to encrypt the data-at-rest. We present an analysis of the security and performance trade-offs in a wide-range of usage and load scenarios.

Using existing known micro benchmarks in devices using encryption without any optimization, we show that encrypted operations can incur negligible overhead for read operations and up to twenty times overhead for write operations for I/O intensive programs. In addition, we quantified the database transaction performance and we observed a 50% operation time slowdown on average when using encryption. We further explore generic and device specific optimizations and gain 10% to 60% performance for different operations reducing the initial cost of encryption. Finally, we show that our approach is easy to install and configure across all Android platforms including mobile phones, tablets, and small notebooks without any user perceivable delay for most of the regular Android applications. Technology trends in both hardware and software have driven the hardware industry towards smaller, faster and more capable mobile hand-held devices that can support a wider-range of functionality and open source operating systems. Mobile hand-held devices are popularly called smart gadgets (e.g. smart phones, tablets, e-book readers).The smart gadget life cycle has evolved drastically in recent years. Nielsen market data trends shows that mobile devices have lifetimes of approximately 6 months between generations. Numerous factors have influenced the industry to grow at this fast pace. One of the most important reasons was the availability of operating systems for mobile handheld devices that were hardware-agnostic by design. These new generations of the smart gadget devices such as the iPhone and Google Android devices are powerful enough to accomplish most of the tasks that previously required a personal computer. Indeed, this newly acquired computing power gave a rise to plethora of applications that attempt to leverage the new hardware.

These include but are not limited to Internet browsing, email, messaging, social networking, and GPS navigation. However, smart gadgets have to come a long way in terms of security. Organizations have come to realize that these commercially available smart gadgets will soon have to serve as an integral part of their operations. This requires a level of security that allows for security of data at-rest and on the move to support secure communications. A major obstacle is that there is a serious lack of National Institute of Standards (NIST) approved encryption algorithms on these commercially available smart gadgets. Much less

common is the existence of any encryption techniques that can pass the strong government validation process in place for any computing device to be used in an adversarial environment. Also, the expectation for each individual application to support encryption runs into the key management problem: other applications in the system can potentially gain access to the key and render the encryption useless. Therefore, there is a need for a practical approach to build common security libraries that operate at the operating system level and provide strong encryption. This system has to be ubiquitous and integrate into the ecosystem of smart gadgets with minimal maintenance and installation cost. Encryption however comes at a significant performance cost. On smart gadgets where resources, like the battery, are very limited, it is important to keep a low footprint on such solutions. In this paper, we focus on analyzing the performance for persistent storage protection using encryption on smart gadget devices. We use a file system encryption which uses certified cryptographic algorithms to store encrypted versions of every file in a source directory. The volume key is decrypted using a password supplied by the user. This is different from full-disk encryption software because the protected data is mounted in memory at a specified mount point in the file system. Moreover, the features and restrictions depend on the underlying partition's file system type.

2.2 Messing with Android's Permission Model

The mobile phone market today performs very well. In many countries, especially in Western Europe and North America, the number of cell phone subscriptions exceeds the population count. Moreover, the number of landlines is decreasing, while the mobile phone sales steadily rose over the past years. According to the Gartner market research handset sales in 2010 have increased by 31.8 percent compared to the year before. With an increase of 72 percent, the Smartphone market grew fastest and is increasingly dominated by the Android operating system. The remarkable history of Android starts in 2005, when Google acquired the 2003-founded startup Android Inc. Until then, only little was known about the young organization's work, whose main business was developing software for mobile handsets. In 2007, Google announced the Open Handset .The Open Handset Alliance at the same time announced the development of Android, which features a complete software platform for mobile handsets including an operating system, middleware and key mobile applications. As the other current Smartphone operating systems (e.g., Microsoft's Windows Phone 7, Apple's iOS), Android allows for the installation of third-party applications by the user. A permission model is used to control the rights granted to installed third party applications (apps). Prior to installing an app, the user is presented with a dialog box showing the permissions required for the app to function fully-featured. The user has to decide either to accept all of the requested permissions or choose not to proceed with the installation. In this paper, we discuss the permission model chosen by the Android Smartphone operating system and present a selection of attacks against it. We show how these attacks can be composed to silently root the targeted Smartphone. In particular, we show how to mount a UI takeover, how to start (malicious) applications after installation without knowledge of the user, how to start applications after boot, and how to obtain a two-way Internet communication for an application that did not request the permission to access the Internet. Putting these attacks together we show how the device can be silently rooted after successfully establishing an Internet connection by downloading a root exploit and running it against the system.

After covering related work on Android security and on attacking Android's permission model in particular, we briefly introduce the Android operating system. We then detail Android's permission model. In Section 5 we present the core part of this paper: our novel attacks against the permission model. The security of the Android platform has attracted the attention of a broad range of security researchers, not only from academia, covering different aspects of Android's security model. For instance, the GTalkSevice connection of Android devices and vulnerabilities in the newly started web-based Android Market were inspected focuses on inter- process communication, while inspects address space layout randomization (ASLR) on mobile devices. A generally valuable overview of current mobile malware in the wild was presented by Felt et al. in 2011.

Besides the comprehensive Android developer documents, various other articles on the web are discussing a broad range of security issues of the Android operating system. Also studies with a general focus on Android exist, e.g., by Enck et al. and Shabtai et al. Other researchers explicitly focus on a single security mechanism, e.g., the construction of Android botnets, and inter-application communication. Other researchers explicitly focus on a single security mechanism, e.g., the permissions used by Android apps. The work of Vidas et al. which discusses the permission model as part of a general description of the Android security model as a whole. In addition, they discuss Android's patch-cycle and the resulting fragmentation in its different versions across the multitude of devices. The patch-cycle turns out to be one of the most important security issues of Android: even though vulnerabilities in the operating system are eventually patched, not all devices will timely be updated, and some will not be updated at all. This is due to the fact that the responsibility for pushing updates to the devices resides at

the device manufacturers. The work provides insight into privilege escalation attacks on the Android operating system. The authors show that both, benign applications exploited at runtime, as well as malicious applications are able to escalated their respectively given permissions. Their work presents an instantiation of a privilege escalation attack exploiting a known vulnerability of the Android Scripting. A more general approach, also involving details on the permission model of the Android operating system, is taken by Hobarth and Mayrhofer. In particular, the authors develop a framework for on-device privilege escalation exploits. They extend their framework such that arbitrary temporary root exploits can be used to gain permanent root permissions.

The research of Shin et al. presents an interesting of the permission model, which exploits the absence of naming rules for permissions. Due to the absence of naming rules, two different permissions with the same name can be in use. Permissions that have once been granted to applications remain in the system, even after removal of the application that originally requested the permission. The authors show that this can lead to the permission being overwritten by another application. Thus, a new application is falsely granted a permission which has been originally granted to another application that is not existent anymore. In their work on inter-process communication in the Android operating system, Chin et al. also discuss the so-called broadcast theft.

They show that malicious applications can easily eavesdrop on public broadcast messages from other applications. These broadcasts are considered implicit intents, which are not protected by sufficiently strong permissions, i.e., System or Signature. Besides elaborating on the implementation of permission enforcement, their work also includes the tool Stowaway which allows mapping API calls of compiled applications to their respective permission. Another study by Enck et al. provides an empirical overview on Android application security and Android malware another interesting study was proposed by Zhou et al. in 2011, in which the authors attempt to detect malware in Android Markets - also by using permission based .The article by Enck et from 2009 general understanding of Android security. We thus refrain from a detailed discussion of the general security framework of Android, but rather recapitulate the permission model as it is the most relevant security feature for this present work.

2.3 Analysis and Research of System Security Based on Android

Wavelet Android is a smart mobile terminal operating platform core on Linux. But due to its open-source software and programmable framework character, it leads the Android system vulnerable to get virus attacks. This paper has deeply researched from the Linux system security mechanism, Android-specific security mechanisms and other protection mechanisms. And on this basis, Android devices have achieved closely guarded on normal state. So that attackers cannot use the kernel module or core library to get highest access permission and be attacked. Meanwhile, to further strengthen the security of Android devices, it enables them to properly handle the high-risk threat. This paper also strengthened intrusion detection system (HIDS) based on the host in order to detect malicious software and strengthen the Android system-level access control. Android is a software stack for mobile devices that includes an operating system, middleware and key applications.

The Android SDK provides the tools and APIs necessary to begin developing applications on the Android platform using the Java programming language. Android is planned to run on many different types of devices. For developers, the range and number of devices means a huge potential audience: the more devices that run Android applications, the more users who can access application. In exchange, however, it also means that applications will have to cope with that same variety of hardware. Android platform is based on Linux technology and composed of operating system, user interface and application components. It allows developer freedom access and modify the source code. It is the free mobile terminal platform with open, the application program equality, no boundaries between applications, facilitate and rapid application development and other advantages. Its issuance breaks monopoly status of the Microsoft Windows Mobile operating system and Nokia's Symbian operating system in the smart mobile telephone platform, while the advantages of its platform also greatly enriched the variety of handheld device software functions. It becomes the intelligent terminal market leader.

Android platform is a set of software package for mobile devices, it includes an operating system, middleware and key applications. Android uses the most innovative characteristic. It allows anyone develop him own applications and freely distributed. But when open provides various conveniences for developers and users, it also increases the safety misery. Due to the lack application development and issuance of effective control, the user is likely downloaded and installed malicious written by software hackers. This will result in some or all of the features in the mobile telephone not working properly. So it deeply studies Android's security mechanisms, it can effectively enhance the protection ability and great significance .Android has built-in tools and support which make it easy for applications to do that,

while at the same time letting the system maintain control of what types of devices application is available to. With a bit of forethought and some minor changes in application's manifest file, it can ensure that users whose devices can't run application will never see it in the Android Market, and will not get in trouble by downloading it. This can explains how it can control which devices have access to its applications, and how to prepare its applications to make sure they reach the right audience. Android provides an open development platform and offers developers the capability to build greatly rich and innovative applications. Developers are free to be superiority of device hardware, access location information, run background service, set alarm, add inform to the status bar, and so on. Developers have full access to the same framework.

The core applications use APIs. The application architecture is designed to simplify the reuse of components; any application can publish its abilities and any other application may then make use of those abilities. This same mechanism permits components to be replaced by the user. From top to bottom Android platform is composed of the Linux kernel, system libraries, Android run time, application framework and so on five parts. Android relies on Linux 2.6 version. It provides core system services: security, memory management, process management, network group, driven model. The core part is equivalent to an abstract level between the hardware layer and other software in the systems, Library and Android Runtime Android includes a set of C/C++ libraries. Various components of Android system are use now. These functions are exposed to developers through the Android application framework.

Android's core libraries provide most 2012 Fifth International Conference on Intelligent Computation Technology and Automation of the function to the Java class libraries. Every Android application runs in its own process, and enjoys the proprietary instance distributed by Dalvik virtual machine, and support multiple virtual machines efficiently run on the same device. These functions can be used to any other application (of course, it is restricted from the framework constraints safety standards); and the same to reuse mechanism, the framework supports component replacement. Android applications are written in Java programming language. The Android platform default includes a set of core applications. It includes home, browser, communication services, contacts and other applications. These applications are written by the Java programming language. It can provide developers a reference. As the Android platform applications equality, developers can write their own applications to replace the default applications provided by Android. The core design idea of Android security architecture is as the following. In the default settings, all applications do not have permission for other applications, systems or users greater impact on the operation. This includes read and write user privacy data (contacts or e-mail), read and write other applications files, access the network or block devices and so on. Android's security mechanism is mainly reflected in two aspects: Android system security and data security. Android system security is referred to the protection of smart terminal itself to operating system. It can prevent unauthorized user external access and authorized service permission. It includes users' behavior detection, operating authority and other measures. The data security is referred to ensure the integrity and legitimacy of stored data, it requires the system can properly transmit data, the authorization process successfully read data. Android system

security protection Android system safety inherited the design of Linux in the design ideology, Android provided security, memory management, process management, network management, drive model and other core service in the kernel. The kernel part is actually an abstract level between hardware abstraction layer and other software group. In practice operation, each Android application runs in its own process. Android system applications are run in some low-level function such as threads and low memory management; Android itself is a separate operating and permission system. In the operating system, each application runs with a unique system identity. (Linux user ID and group ID). Each parts of the system were also using their own independent identification mode. The most security functions of the system are provided by the permission mechanism. Permission can be restricted to particular specific process operations, and can also restrict URL permission to access specific data segment.

2.4 The Design & Implementation Of Android File Access Control System

Android is a popular operating system on mobile devices, and people care about the security issues of Android very much. Based on the analysis of the defects in Android security mechanism, this paper proposes the design and implementation of an Android File Access Control System that supplies authorization and authentication to the file operations in order to prevent the sensitive files. The simulation results indicate that the Android File Access Control System achieves the goal of file access control on Android. This requires a level of security that allows for security of data at-rest and on the move to support secure communications. A major obstacle is that there is a serious lack of National Institute of Standards (NIST) approved encryption algorithms on these commercially available smart gadgets. Much less common is the existence of any encryption techniques

that can pass the strong government validation process in place for any computing device to be used in an adversarial environment. Also, the expectation for each individual application to support encryption runs into the key management problem: other applications in the system can potentially gain access to the key and render the encryption useless. Putting these attacks together we show how the device can be silently rooted after successfully establishing an Internet connection by downloading a root exploit and running it against the system. After covering related work on Android security and on attacking Android's permission

model in particular, we briefly introduce the Android operating system. We then detail Android's permission model. In Section 5 we present the core part of this paper: our novel attacks against the permission model. 2 Related Work The security of the Android platform has attracted the attention of a broad range of security researchers, not only from academia, covering die rent aspects of Android's security model. For instance, the GTalkSevice connection of Android devices and vulnerabilities in the newly started web-based Android Market was inspected. Focuses on inter-process communication, while inspects address space layout randomization (ASLR) on mobile devices. This will result in some or all of the features in the mobile telephone not working properly. So it deeply studies Android's security mechanisms, it can effectively enhance the protection ability and great significance.

Android has built-in tools and support which make it easy for applications to do that, while at the same time letting the system maintain control of what types of devices application is available to. With a bit of forethought and some minor changes in application's manifest file, it can ensure that users whose devices can't run application will never see it in the Android Market, and will not get in trouble by downloading it. This can explains how it can control which devices have access to its applications, and how to prepare its applications to make sure they reach the right audience. Android provides an open development platform and offers developers the capability to build greatly rich and innovative applications. Developers are free to be superiority of device hardware, access location information, run background service, set alarm, add inform to the status bar, and so on. Developers have full access to the same framework. The core applications use APIs. The application architecture is designed to simplify the reuse of components; any application can publish its abilities and any other application may then make use of those abilities. This same mechanism permits components to be replaced by the user. From top to bottom Android platform is composed of the Linux kernel, system libraries, Android run time, application framework and so on five parts.

Android relies on Linux 2.6 version. It provides core system services: security, memory management, process management, network group, driven model. The core part is equivalent to an abstract level between the hardware layer and other software in the systems, Library and Android Runtime Android includes a set of C/C++ libraries. Various components of Android system are use now. Every Android application runs in its own process, and enjoys the proprietary instance distributed by Dalvik virtual machine, and support multiple virtual machines efficiently run on the same device. These functions can be used to any other application (of course, it is restricted from the framework constraints safety standards); and the same to reuse mechanism, the framework supports component replacement.

Android applications are written in Java programming language. The Android platform default includes a set of core applications. It includes home, browser, communication services, contacts and other applications. These applications are written in the Java programming language. It can provide developers a reference. As the Android platform applications equality, developers can write their own applications to replace the default applications provided by Android.

The core design idea of Android security architecture is as the following. In the default settings, all applications do not have permission for other applications, systems or users greater impact on the operation. This includes read and write user privacy data (contacts or e-mail), read and write other applications files, access the network or block devices and so on. Android's security mechanism is mainly reflected in two aspects: Android system security and data security. Android system security is referred to the protection of smart terminal itself to operating system.

Paper 1 : **The Design and Implementation of Android File Access Control System**	• **Implements a File Access Control System** • **All applications would require specific permissions.**
Paper 2: **Implementing and Optimizing an Encryption File system on Android**	• **Implements an Encrypted file structure.** • **Require decryption of files on each access.** • **No caching.**

3. Application Design

The application "Rebin" intercepts the deletion of files by user or other applications and instead moves the said files to an encrypted location in the secondary memory of the phone. A simple implementation would simply move the files to a hidden location. An advanced version would encrypt the file using the user's password and writes it directly to the memory bypassing the File Allocation Table and use an unallocated location which is also then locked from being used by other applications. Also the files are renamed to prevent file name collisions.

A log is created for each intercepted deletion request which contains the original location and the memory location to which it is written, along with the file size. The file sizes are incremented to prevent memory over flow and the operation is stopped when more than 10% of the total available size is used by the App.At that point, the user is reminded that the deleted file size has exceeded the limit. Then the user can either increase the limit or delete the files from Recycle Bin. Also whenever certain parameters are met, deletion of files from Rebin is done. For example, a file deleted more than a month ago or file over 1GB memory size will be deleted. Deleted files can be recovered using the interface provided by the application which uses the log to sort either using name, file modification and deletion or file size. User can select the files and Rebin will rename the file after decrypting and move it to the original location.

4. Conclusion

Mobile phones have become a necessity in our daily life and have chances for unexpected loss of data. Our project "Rebin" can efficiently restore the data that has been deleted with just a tap of it.

References

[1] C. Wang, W. Duan, J. Z. Ma, and C. H. Wang, "The research of Android system architecture and application programming," Computer Science and Network Technology (ICCSNT 11), IEEE Press, Dec. 2011, pp. 785-790, doi:10.1109/ICCSNT.2011.6182081.

[2] W. Tang, G. Jin, J. M. He, and X. L. Jiang, "Extending Android security enforcement with a security distance model," Internet Technology and Applications (iTAP 11), IEEE Press, Aug. 2011, pp. 1-4, doi:10.1109/ITAP. 2011.6.