# DroidSearch: A Search Engine to Automatically Collect, Disassemble, Index and Search Android Malware

Mingshen Sun, Min Zheng
{mssun, mzheng}@cse.cuhk.edu.hk
Group 10
1155021091 & 1155006998

## ABSTRACT

With the increasing number of smartphone users, hacker have moved there targets to smartphone. Recently, a lot of malware pop up into analysts' eyes. Different anti-virus company have paid attention to this area. Many malware analysts are working on the malware disassembling and analyzing. To improve the effectiveness of analyzing malware, DroidSearch give a practical solution. DroidSearch can automatically collect, disassemble, index and search Android malware. By developing a extensible third-party markets application crawler, we have downlaoded 90,368 applications including 1,688 malware. In addition, DroidSearch can disassemble the applications to source codes and index them into search engine. Then analysts can use DroidSerch to perform full-text search, definition search or symbol search. Also, DroidSearch can give term suggestions for analysts when they type the wrong term. We successfully uncovered the remote server of the malware and show the effectivness of DroidSearch.

## Keywords

Search Engine, Malware, Smartphone Security

## Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval–information filtering, search process, digital libraries

## 1. INTRODUCTION

Smartphones and mobile devices are rapidly becoming the dominant and indispensable devices for many users. In the report of Gartner [13], in fourth quater of 2011, the growing number of smartphone sales around world is 47 percent. A report from nielsen [21] show that the amount of smartphone in stock takes more than half of all mobile phones. IDC [18] forecasts the scale of the smartphone will reach one billion until the year of 2015. Unfortunately, these devices are usually resource constraint and it makes malware easily camouflage and sabotage personal information. Juniper Networks Mobile Threat Center [19] reports that they have detected 155% growing in smartphone malware in all different platforms as compared with 2011. Although there are number anti-virus products for the PC-based systems, but mobile devices and mobile malware have many unique characteristics that make traditional anti-virus software ineffective. For instance, mobile malware can easily generate new variants or signatures that can bypass the detection of traditional anti-virus system.

Facing with these so many malware in smartphone platform, analysts find two effective methods to tackle the problem and analysis malware samples. The first one is static analysis. Researchers should manually disassemble the application and discern the structure of the reverse engineering source codes. By checking the reverse source code, analysts can easily find the malicious code and determine if this application is a suspicious malware. There are some systems aim to give a better analysis environment like [16, 25, 26, 22]. However, researchers should face with another hurdles, because of the obfuscation of malware, it is difficult to screen the codes manually. Sometimes, some of the malicious payloads may be downloaded from Internet when the running of the applications. Therefore, dynamic analysis method wants to solve this problem. This can dynamically track the information leakage or the malicious behavior which cannot be analyzed from source code. Some other systems like [24, 12, 23, 11] shows the effectiveness of the dyanmiac analysis.

In recent analysis practice, static analysis is still an important method. There are some hurdles which analysts confront when they conduct experiments with static analysis with thousands of suspicious malware samples.

1. There are hosts of malware samples in the database, how to quickly and easily find out a malicious sample that is related to the research. By checking the source codes of the suspicious applications, analysts can determine which contains malicious segments.

2. Researchers have to discern the malware in source code level to figure out the malicious code fragments and structures. The malicious codes always have some obvious strings or signatures like "readSmsMessage()" or "http://adrd.xyz8**.com".

3. How to orgnize the malware database well and make it easy to retrieve information and help analysts reverse

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Copyright 20XX ACM X-XXXXX-XX-X/XX/XX ...$15.00.

engineer these malware? From the reverse engineering codes, we can analyze the malicious behavior, but also retrieve some other important information. For example, we can figure out the URL address of the remote server from which malware can download updated packages or some premium phone number which can be used to send premium SMS messages.

To address these problems mentioned above, we propose DroidSearch, a search engine to automatically collect, disassemble, index and search Android malware. DroidSearch can automatically collect malware samples from third-party markets. Furthermore, DroidSearch will disassemble the applications to source code (smali code) and then indexes these codes in to search engine. At last, analysts can use it to perform full text search to retrieve important information from the reverse engineering codes easily. The contributions of our system are:

- We develop a automatic crawler to collect malware samples from third-party markets. We have crawled 90,368 applications about 268.13GB from ten third-party markets, one malware share forum 1.
- We develop a user-friendly system called DroidSearch which can index disassembled source codes. It will automatically transform the Android applications to smali codes which is the reverse engineering source codes.
- DroidSearch also have full text search, definition search and symbol search features. Analysts can use this features to search the information from the disassembled codes.
- DroidSearch can generate cross reference between source codes of applications. Analysts can easily jump from the calling method to the definition of the method.
- We write the syntax highlighter for the disassembled code (smali code). It provides a user-friendly interface for analysts to quickly figure out the significant information.

This paper is organized as follows. Section 2 give some background knowlodge of Android malware analysis, and shows the importance of third-party markets of Android applications. Section 3.2 illustrates the features, functionalities and implementations of each components of the system. Also, we will show a example to perform analysis on malware source codes. Section **??** evaluate the performance of DroidSearch and show some statistics when implementing the system. Section 5 provides the plan of future work and compare the difference among several other related works. At last Section 6 concludes the contributions of DroidSearch.

# 2. BACKGROUND

In this section, we briefly present the process to preform a static analysis for Android malware and we will introduce the importance of Android third-party markets.

## 2.1 Android Malware Analysis

In this subsection, we will introduce the basic malware analysis methods, especially static analysis and then give you a example in practice.

Android malware is same like malware in PC. When user run an application in his smartphone, it will trigger the malicious behaviors. Therefore, if the analysts want to see the malicious behaviors of the malware, he should run this application in emulator or the real phone. However, we need to manually trigger the malicious code first. But analysts may not know which button or which activities will cause the malicious behaviors. That's why it is important for analysts to do the static analysis.

Before understanding the process of static analysis, we should know some basic knowledge of Android applications. Figure 2.1 illustrate the basic building process. Android applications are different from normal program like PE application in Windows or applications on iOS. Android applications are written by Java and compiled by Google's Android Development Toolkit (ADK). That means that when we write an application, there should be many .java files in the project. And then, compiler will transform them to .class files. For Android, these .class file will pack into one file classes.dex. After compiling these files, Android Asset Packaging Tool (aapt) will package all the resource file like images and audios together with the classes.dex file into one file (.apk file). This file contains all the required file when running the applications. Actually, this file is just a zip package which it is easy for us to unpack the file into several resource file. Meanwhile, we can reverse engineer the classes.dex file to the source codes. This is the basic method which analysts use to analyze the malware and figure out malicious codes.

Let us now discuss the process of static analysis in Android application. For example, if an analysts receive a suspicious malware. Firstly, the analyst will use disassemble tools (smali/baksmali [9], dex2jar [5], apktool [4], ..., etc) to reverse engineer the classes.dex file which contains the information about the application. After disassembling the application, there will be several source file which are similar with the code writer develop before. This file is .smali file which have different syntax with Java, but it can easily present the classes and methods or API calling in the program. The analyst get these file and then discern the file one by one to see if some lines contain dangerous API call like READ_SMS_MESSAGE. Also, analyst can find out the remote server address of malware. This malicious remote server is an important signature for a kind of malware from same family. However, this kinds of manually checking takes a lot of time and the effectiveness is very poor.

## 2.2 Third-party Markets

In this subsection, we will illustrate that the spread of Android malware in the third-party markets.

Application Market is a place from which smartphone users can download games, utilities and other applications. Different vendors provide their official sites for users to search and download applications. These include Apple's App Store [10], Google Play Store [15] and Microsoft Marketplace [20]. Due to the instability of Android Official Market (Google Play) in China, Chinese user always download the applications from third-party markets. Because there is not any sensor or few screening on the submissions of applications, there will be a lot of malware hidden in the third-party markets. Therefore, we should pay more attention to the third-party markets on the malware. For example, the hacker can easily write a malware and then publish in the third-party

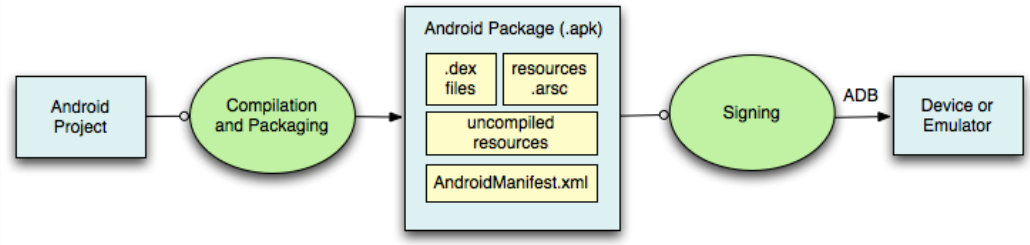


**Figure 1: Building Process of Android Application[14]**

markets but no one could notice the malware.

# 3. SYSTEM DESIGN

In this section, we will show the design goals when developing DroidSearch. Moreover, we will introduce each component of the system and explain the functionalities and implementations.

## 3.1 Design Goals

DroidSearch is developed to meet the following goals:

- Capability of Application Crawling. The system should have an extensible crawler to collect applications in the third-party markets. It should keep updating the application repository dealing with the increasing number of application which may contain some malware.
- Capability of Android Application Disassembling. Because the orignal applications are assemble codes which is unreadable, the system should reverse engineer the application to human readable codes (smali codes). Within these source codes, analysts can discern the malicous behvior from them. The system can use this data to perform search feature to users.
- Capability of Indexing Android Reverse Engineering Source Code. Different from unstructured document, smali code has strict syntax. Therefore, the system should process this document and index the information.
- Capability of Retrieving Malicious Information. The malicious codes such as the address of remote server always hide behind the complicated codes. The system should have a user-friendly interface for analysts to retrieve these malicious information conceal in the reverse engineering codes.

## 3.2 Building Blocks

Figure 3.2 depicts the archetecture of DroidSearch. There are four components in DroidSearch respoinsible for different tasks. The first component is crawler which will collect the url addresses of the applications in the third-party markets. And then the crawler will automatically store these urls into the database. The second component is downloader. Downloader can download the application from the third-party markets into application repository. The third component is indexer which is responsible for reverse engineering the application to source code (smali code). After transforming the applications, indexer will analyze the structured data and then store for search engine.

### *3.2.1 Extensible Crawler*

In DroidSearch, we implement an extensible crawler based on Scrapy [7]. Scrapy is an open-source crawler platform written by Python. Because there are so many third-party markets, we require a crawler which can handle different markets and crawl the address of applications.

Figure 3.2.1 depicts the architecture of our crawler. The Scrapy engine is responsible for controlling the data flow between other components of DroidSearch. Scheduler is a queue for engine when engine requests the urls. The main task of downloader is to fetch the web pages. Spiders are custom classes in which we add the third-party markets' urls. Moreover, we can set the selector to select item. That means that we can determine which part of the pages should be selected and pass to item pipeline to perform further handling. In DroidSearch, we use XPath to analyze HTML web pages and select the url addresses of applications. Figure 3.2.1 shows the XPath of the url address of one application. Therefore, we use this XPath to select all the addresses from this website of thid-party market. Because, developers of third-party markets' websites always have the same templates and the adresses of applications are always in the same place. Also you can decide which domains will be crawled and which domains will be ignore in order to specify the crawling scope. Item pipeline can process the items passed from spiders. For example, in DroidSearch, we write a pipeline can store the crawled applications' urls into database (SQLite). Download middlewares are some hooks between engine and downloader when something passing from engine to downloader. Spider middlewares are hooks between engine and spiders. So we can extend the middleware and add the proxies for our crawler to adapt to the environment behind proxy.

In practice, the workflow of crawling a third-party market is following:

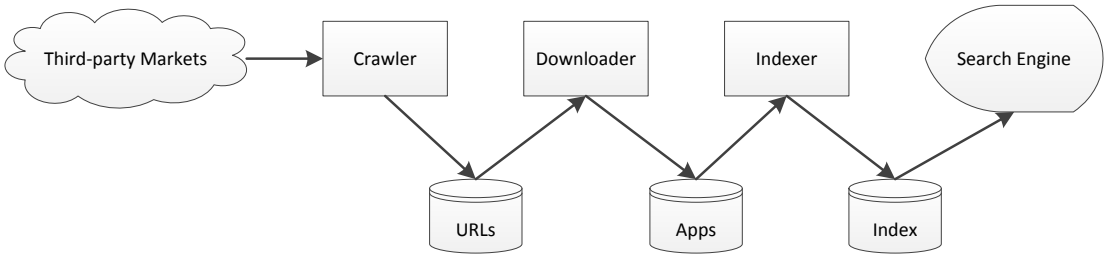


**Figure 2: Building Blocks of DroidSearch**

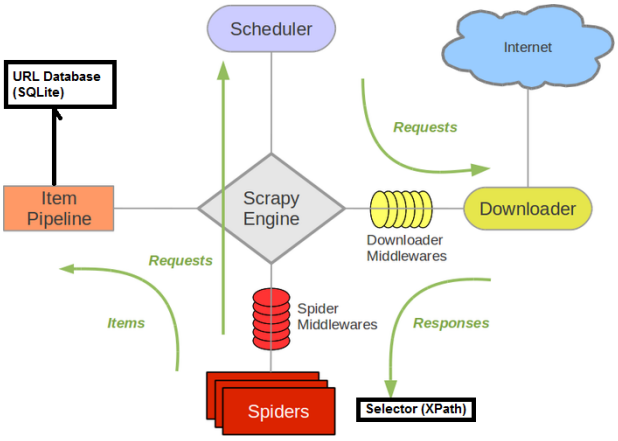


**Figure 3: Architecture Of Crawler (based on [8])**

1. Crawling engine pick a seeds (a.k.a., the url of third-party market) and give this address to scheduler.

2. Scheduler receive the address and then request downloader to download the whole pages from internet. The results will pass to spiders.

3. Spider hold the page and analyze it using XPath. The item which is the url address of application will be filtered and delivered to item pipeline. Furthermore, spider will give the hyperlinks in the page to engine in order to download the whole websites.

4. Crawler engine keep crawling the pages until there is nothing in crawling queue.

By using this kind of architecture and running process, we can easily implement other third-party markets into our crawler. Also, the crawler can keep running and update the database automatically.

### *3.2.2 Robust Downloader*


**Figure 4: XPath of An Application[14]**

After we getting a number of application downloading addresses, downloader will start to download this urls from Internet. However, the environment of Internet is not as perfect as we can see. There will be some dead links, timeout or page not found. In addition, some thrid-party markets website will constrain the number of connections. So it is not easy to build a downloader which can handle these bad web environment. Our downloader is written to meet these problems. The downloading process is as follow:

- Firstly, downloader will pick a url address from database and judge if this application has been already downloaded.

- Secondly, it will request the address and download the application into application repository. Note that the downloader can handle the exception when downloading applications. And it will drop the address which is dead link, and restart download the application when there is some interruption in downloading.

### 3.2.3 Effective Indexer

There are two parts in indexer. The first part is to disassemble the applications to source codes (smali codes). In addition, disassembling is a time consuming work. We should make an effective tools to transform the applications to source codes. The second part is to index the source codes in order to give search engine a metadata which can be easily searched. Also, The syntax highlight is based on the second part.

There are so many reverse engineering tools for our system. We choose to implement smali/baksmali into DroidSearch. The main function of smali/baksmali is to analyze the bytecode of Android (classes.dex) and then constructs a smali file which contains human readable source codes. This codes is a little different from Java, the original writing programming language of Android. The syntax of smali is loosely based on Jasmin's/dedexer's syntax, and supports the full functionality (annotations, debug information, line information, etc) of the dex format (Android bytecode.) Additionally, Android application (APK file) is a zip file which contains the resources and classes.dex which is the bytecode of Android application. Therefore disassembling the Android application take these two steps:

1. The system should unzip the APK file and extract the bytecode file from the package.
2. smali/baksmali will disassemble the classes.dex into several smali files. Each file contain the source codes of a class in the application.

At last, we get the source codes of Android application in smali syntax.

After reversing the applications to smali source code. DroidSearch should index these source code and extract the classes and methed in the source code. This indexer is based on OpenGrok [6] and Lucene [2]. However, both of them do not support the smali syntax. So we should write our own syntax indexer. The smali code is structured as the codes. The method begins with ".method" and ends with ".end method". So we can treat this part as a method. Moreover, "invoke-virtual" is a key word which can invoke other methods. And "const-string" can claim a string object and initialize as "Hello World!". So we construct a key words list and a syntax regulation for Lucene. And then write a analyzer to index this smali code. Therefore we can separate the methods, symbols and calling in the smali code. That provide a convenient metadata for search engine to search by method name or by other symbols.

```
1  .class public LHelloWorld;
2
3  .super Ljava/lang/Object;
4
5  .method public static main([Ljava/lang/String;)V
6      .registers 2
7
8      sget-object v0, Ljava/lang/System;->out:Ljava/io/
           PrintStream;
9
10     const-string v1, "Hello World!"
11
12     invoke-virtual {v0, v1}, Ljava/io/PrintStream;->
           println(Ljava/lang/String;)V
13
14     return-void
15
16 .end method
```

### 3.2.4 User-friendly Interface

The most important component of DroidSearch is the search engine and a user-friendly interface. DroidSearch uses Lucene as the fundamental library of search engine. Apache Lucene is Java based full-feature text search engine library. There so many useful library for constructing our system. In addition, we provide a web-based service for analysts search the source code online. JSP is a perfect technology to write a web service. And we write the web interface by using JSP and running the web application on Tomcat [3] which is a web service provider. Above the features of search engine of DroidSearch is following:

- Full-feature text search. With DroidSearch analysts can search anything they want. Such as definition search, symbol search. Even the user can combine different search fields.
- It supports wildcard (* and ?) and boolean operators (&&, || and !). These can help analysts to search the best results.
- Suggestion. Like normal search engine, DroidSearch provide a suggestion feature to analysts. When analysts type a wrong search term to the engine, DroidSearch will automatically provide several suggested terms for analysts to select. Furthermore, all of the term are from the indexed documents (source codes).
- Relevant feedback. we aim to provide a relevant feedback to analysts. If analysts select some source codes as relevant document, DroidSearch will rerank the results for analysts. Because a comprehensive relevant feedback features require a lot of works. Therefore, we plan to add more features on this area in the future.
- Cross reference in source code. Due to the complexity of the calling between different classes and methods. A cross reference can reduce a lot of search work than usual. For example, when an analyst interests with a method which may contain dangerous API calls. He can click the method directly and the page will redirect to the definition of this method or the implementation codes of the method. The cross reference function is based on exuberant Ctags. Ctags can generate an index or tag file of language objects found in source file. By using ctags users can quickly and easily locate the position of items by a text editor or other utility.
- Syntax highlight. For the program analysts, syntax highlight is a significant feature. Because smali syntax is different from other usuall codes syntax. We write our own smali syntax for users. By using the highlight, analysts can easily find the methods calling positions and also the variable claiming.

By using Apache Lucene, we can simply implement the full-text search, defination search and symbol search. The SpellChecker in `org.apache.lucene.search.spell.SpellChecker` package provide a useful API to perform spelling check in DroidSearch. We can use the Lucene API open the file system directory of each spelling index and then find the suggestion results using SpellChecker class. Furthermore, syntax highlight can be implemented by using regular expression. regular expression is a filtering methodology to

compute the matching strings of text concisely and flexibly, such as particular characters, words, or patterns of characters. For example "$[\backslash t\backslash f]+$" represents white space in the documents and "$EOL = \backslash r|\backslash n|\backslash r\backslash n$" represents the end of the line. By using this, we can divide the source codes into different part representing different meanings.

By designing and implementation of each component of DroidSearch, we get a full functionality search system which can disassemble Android application, retrieve information from Android reverse engineering source codes. Figure 3.2.4 shows the homepage of DroidSearch, the analysts can type the search items to search source codes from diverse documents. Figure 3.2.4 illustrates the result of searching "adrd" which is a malware family name. we can see from the result that there are three documents (classes) contain this key word. And we can click the document and see how did this malware operate on the "adrd". After analyzing, we know that the url contain "adrd" is an address of remote server. And also DroidSearch provide the interface to give several term suggestion when you type the wrong terms. Figure 3.2.4 shows the feature of syntax highlight. It is obvious to notic analysts the position of methods definition or the strings in the source code. Figure 3.2.4 shows the suggetions feature of DroidSearch. When user type "downlo" which does not exsit in the documents, DroidSerch will provide several term candidates for users to choose.

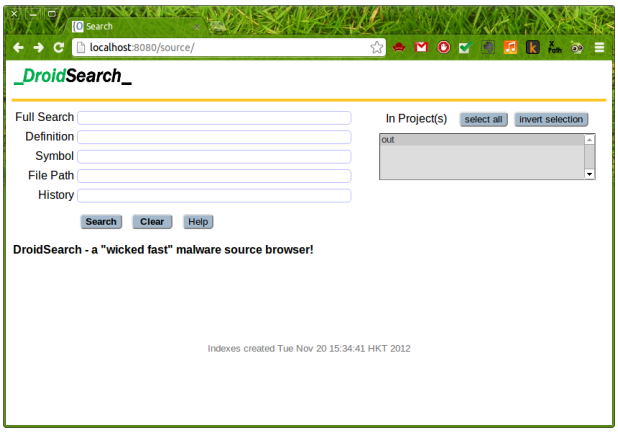


**Figure 5: Homepage of DroidSearch**

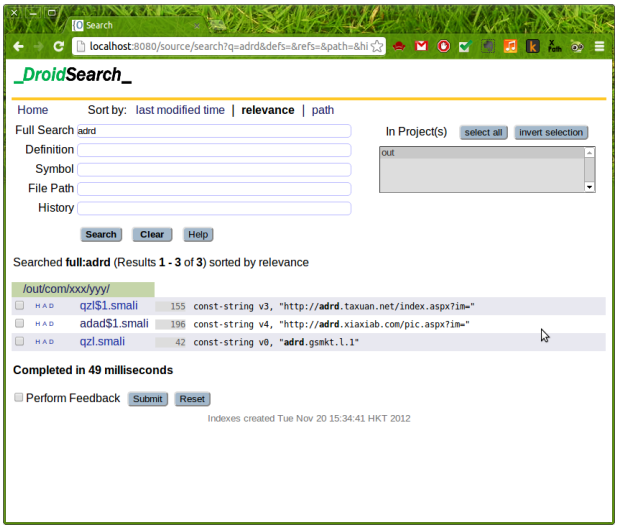


**Figure 6: Search Result of DroidSearch**

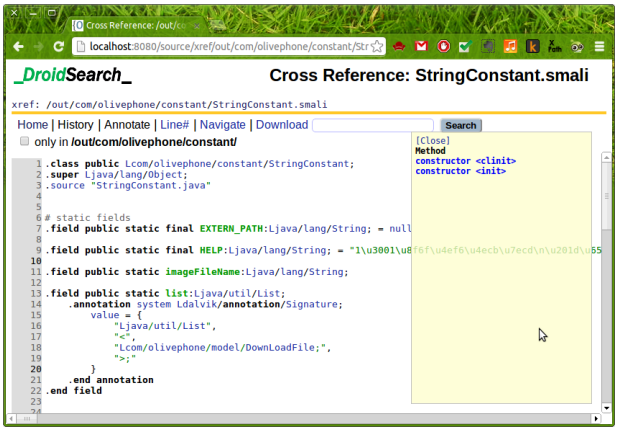


**Figure 7: Demostration of Syntax Highlight of DroidSearch**

# 4. EVALUATION

In this section, we evalute the performance of crawler and search engine and provide some statistics of DroidSearch.

We implement the crawler for several months. Because crawling waste a lot of time, so we construct a server to keep downloading the applications. On average, we can crawl a whole third-party market and download the applications for several days. Basically, the time is based on the condition of internet connection and the exception for specific third-party markets. Because some third-pary markets restrict the connection time and will ban the crawler which keep crawling the web pages. Until now, we have crawled 9 third-party markets and one official marketplace and a blogsite which post the latest malware samples. Table 1 shows the targets we crawl until now. The totle size of the application is about 268.13GB.

Given these applications, DroidSearch will perform indexing. It will take almost five seconds to index an application on average. However, different application may have various size. The running time depends on the size of the applications. The procedure contains disassembling the application into several smali codes and indexing the source codes into metadata for search engine.

For search engine, we invite the analysts from anti-virus company to evalute the effectiveness of DroidSearch. They give a high evaluation of the functionalities of DroidSearch. Furthermore, they said this web-based service really provide some conveniece for analyzing malware source codes. That reduce the analysis time when searching the information from lots of malware source codes.

# 5. FUTURE WORK & RELATED WORK

DroidSearch is a search engine to give search service for

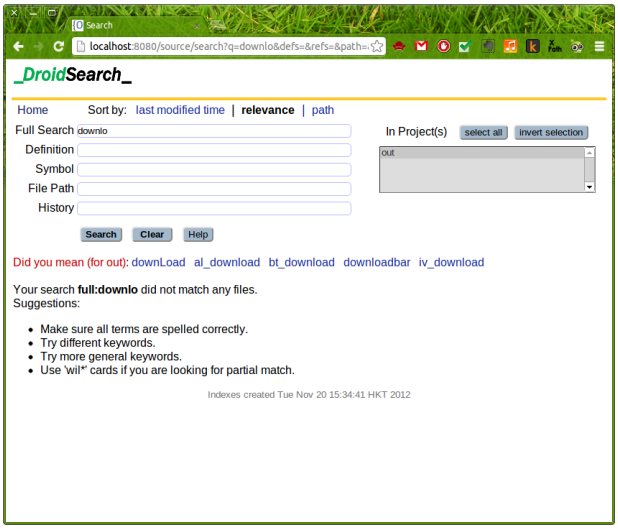


**Figure 8: Suggestion of DroidSearch**

| Marketplace | Samples | Size |
|---|---|---|
| Google Play | 23164 | 41.2GB |
| www.appchina.com | 30682 | 87.2GB |
| www.souapp.com | 3718 | 10.4GB |
| www.appsapk.com | 620 | 0.58GB |
| www.apkxyx.com | 7399 | 41.5GB |
| www.apkke.com | 6678 | 15.5GB |
| www.angeeks.com | 2449 | 8.61GB |
| www.520apk.com | 8782 | 28.6GB |
| android.d.cn | 5939 | 32.1GB |
| www.apkla.com | 557 | 1.94GB |
| contagiominidump.blogspot.com | 184 | 0.18GB |
| Other | 196 | 0.32GB |
| All | 90368 | 268.13GB |

**Table 1: Sources where DroidAnalytics obtains application samples**

malware analysts. With the data of the search engine keeping increasing, the search results will be a problem for analysts. Therefore, implementing a better ranking algorithm is a good idea for malware researchers. We plan to implement a ranking alogrithm by analysts' feedback. That means that the source codes with malicious codes will rank the higher position.

There are two project should be mentioned which is same as our DroidSearch. The first one is from industry, [1] is also a web-based analysis project. However, this project mainly focus on the organization of a malware source codes. And also like our system provides a user-friendly interface. In addition, the search feature in that system can only perform full-text search. The second one if from academy, [17] is a system to scalably analyze the dangerous behavior of a particular application. This system can give a rank of lots of applications and figure out which application may have malicous codes. This work is based on heuristic analysis on malware samples. It is different from our purpose to provide a effective interface for analysts to analyze Android malware.

# 6. CONCLUSION

DroidSearch is a search engine which can automatically collect, disassemble, index and search Android malware. By developing a extensible third-party markets application crawler, we have downlaoded 90,368 applications including 1,688 malware. In addition, DroidSearch can disassemble the applications to source codes and index them into search engine. Then analysts can use DroidSerch to perform full-text search, definition search or symbol search. Also, DroidSearch can give term suggestions for analysts when they type the wrong term. We successfully uncovered the remote server of the malware and show the effectivness of DroidSearch. At last we give some future works to improve the project and provide a better service.

# 7. REFERENCES


[1] Dexter. `http://dexter.dexlabs.org/s`.

[2] Apache Lucene. `http://lucene.apache.org/core/`, 2012.

[3] Apache Tomcat. `http://tomcat.apache.org/`, 2012.

[4] apktool. `http://code.google.com/p/android-apktool/`, 2012.

[5] dex2jar. `http://code.google.com/p/dex2jar/`, 2012.

[6] OpenGrok. `http://hub.opensolaris.org/bin/view/Project+opengrok/`, 2012.

[7] Scrapy. `http://www.scrapy.org/`, 2012.

[8] Scrapy Architecture. `http://doc.scrapy.org/en/0.16/topics/architecture.html`, 2012.

[9] smali/baksmali. `http://code.google.com/p/smali/`, 2012.

[10] Apple Inc. Apple App Store. `http://www.apple.com/iphone/from-the-app-store/`.

[11] Burguera, I., Zurutuza, U., and Nadjm-Tehrani, S. Crowdroid: behavior-based malware detection system for android. In *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices* (New York, NY, USA, 2011), SPSM '11, ACM, pp. 15–26.

[12] Enck, W., Gilbert, P., Chun, B.-G., Cox, L. P., Jung, J., McDaniel, P., and Sheth, A. N. Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. In *Proceedings of the 9th USENIX conference on Operating systems design and implementation* (2010).

[13] Gartner. Gartner Says Worldwide Smartphone Sales Soared in Fourth Quater of 2011 With 47 Percent Growth. http://www.gartner.com/it/page.jsp?id=1924314, 2012.

[14] Google. Building Process of Android Applications. http://developer.android.com/tools/building/index.html, 2012.

[15] Google Inc. Google Play Store. https://play.google.com/store.

[16] Grace, M., Zhou, Y., Wang, Z., and Jiang, X. Systematic Detection of Capability Leaks in Stock Android Smartphones. In *Proceedings of the 19th Annual Network & Distributed System Security Symposium* (2012).

[17] Grace, M., Zhou, Y., Zhang, Q., Zou, S., and Jiang, X. Riskranker: scalable and accurate zero-day android malware detection. In *Proceedings of the 10th international conference on Mobile systems, applications, and services* (New York, NY, USA, 2012), MobiSys '12, ACM, pp. 281–294.

[18] IDC. IDC Worldwide Quarterly Mobile Phone Tracker. Tech. rep., 2011.

[19] Juniper. 2011 Mobile Threats Report. Tech. rep., 2012.

[20] Microsoft Corporation. Windows Phone Marketplace. http://www.windowsphone.com/en-US/marketplace.

[21] Nielsen. Smartphones Account for Half of all Mobile Phones, Dominate New Phone Purchases in the US. Tech. rep., 2012.

[22] Xu, R., Saïdi, H., and Anderson, R. Aurasium: practical policy enforcement for android applications. In *Proceedings of the 21st USENIX conference on Security symposium* (Berkeley, CA, USA, 2012), Security'12, USENIX Association, pp. 27–27.

[23] Yan, L. K., and Yin, H. Droidscope: seamlessly reconstructing the os and dalvik semantic views for dynamic android malware analysis. In *Proceedings of the 21st USENIX conference on Security symposium* (2012).

[24] Zheng, C., Zhu, S., Dai, S., Gu, G., Gong, X., Han, X., and Zou, W. SmartDroid: an Automatic System for Revealing UI-based Trigger Conditions in Android Applications. In *Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices* (2012).

[25] Zheng, M., Lee, P. P. C., and Lui, J. C. S. ADAM: An Automatic and Extensible Platform to Stress Test Android Anti-Virus Systems. In *Proceedings of the 9th Conference on Detection of Intrusions and Malware & Vulnerability Assessment* (2012).

[26] Zhou, Y., Wang, Z., Zhou, W., and Jiang, X. Hey, You, Get Off of My Market: Detecting Malicious Apps in Official and Alternative Android Markets. In *Proceedings of the 19th Annual Network & Distributed System Security Symposium* (2012).