University of California

Los Angeles

Retargeting Attacks from Windows to Android With Limited Data Extraction

A thesis submitted in partial satisfaction

of the requirements for the degree

Master of Science in Computer Science

by

Cary G Ng

2014

Abstract of the Thesis

Retargeting Attacks from Windows to Android With Limited Data Extraction

by

Cary G Ng

Master of Science in Computer Science

University of California, Los Angeles, 2014

Professor Miodrag Potkonjak, Chair

Targeted attacks against networks of Windows machines have become a serious problem in many countries, as they are now being conducted by professionals for purposes of data theft, sabotage and espionage. All of the known attacks that have been thoroughly studied so far mainly involve the Windows operating system, for the reason that most systems used to control infrastructure are on Windows. However, it is very likely that targeted attacks will evolve towards Android devices, as we notice more people using smartphones to facilitate their daily activities especially in corporate environments. This paper will illustrate in detail how a targeted attack would occur on an Android environment. It will also provide detailed analysis on a scientific approach used for limited data extraction given the resource constraints on an Android environment.

Table of Contents

List of Figures

List of Tables

Acknowledgments

First of all, I would like to thank my advisor, Professor Miodrag Potkonjak, for allowing me to reach my fullest potential during my graduate studies. His continuous encouragement and supportive advices have been very helpful in the completion of my research. I would also like to thank Professor Mario Gerla and Professor Carlo Zaniolo for their good feedback and advices on my research. Also, I want to thank both Steve and Craig for helping me every step of the way with all of the administrative issues. Special acknowledgements as well to my parents and Tinnie for motivating me to complete my graduate studies. Finally, I want to thank Carson, Mathea, Haidee and Lester for always taking their time to listen to my research ideas and discoveries.

CHAPTER 1

Introduction

In this chapter, we will provide a brief overview on how malware attacks have evolved over time. We will also show the current popularity of Android devices. We will then explain what a targeted attack is, how each security vendor defines a targeted attack, and provide a few examples of highly publicized targeted attacks on Windows. A brief overview on existing literature related about file/directory arrangement on Windows will also be provided. Afterwards, we will state our research objectives and practical applications that can be derived from this research.

1.1 Brief History of Malware and Targeted Attacks

In the 1990s, malware such as viruses and Trojan horses were mainly created for fun and as a hobby. A few of the malware families may have had destructive payloads such as destroying data on hard drives, but nothing more complex than that. However, in the early 2000s, some malware families started allowing spammers to use infected terminals as SMTP proxy relays as a paid service. This was evident in malware families such as Fizzer and Bagle [Pro11]. It is obvious here that malware has evolved from simply being a hobby to more of a revenue generating activity. This is when professional hackers and criminals started becoming involved in the industry. As cybercrime becomes more lucrative, its technology would naturally evolve. Nowadays, malware has become so sophisticated that it is now a major part of sophisticated targeted attacks used for major cyber warfare, espionage and theft, especially a type of attack called advanced persistent threats. The sophistication of these malwares allow the attackers to focus more on the intelligence and planning aspect of a targeted attack as opposed to trying to manually construct exploits for newly discovered vulnerabilities.

1.2 The Popularity of Android Devices

Android devices have become the most popular handheld device worldwide. According to a report by IDC in 2013, the ratio of smartphones vs PCs is more than 3 to 1 (65.1% vs 20.2%), and is expected to grow even further [Idc13]. Among these smartphones, a report published by Gartner in 2013, Android sold over 78% of these smartphones during that same year [RM14]. In fact, malware writers have already predicted the dominance of Android smartphones a long time ago. More than 98% of existing malware for mobile platforms is for Android. Although the total number of malware samples seen each day for Android is still not as significant compared to the ones on Windows, the trend on Android malware samples is still expected to grow. [FG13]

1.3 Role of Malware in Targeted Attacks

Malware is now used by professional hackers to assist them with their targeted attacks. In fact, malware plays a major role by allowing the hacker to obtain full control of the infected terminal. Many instances of malware infections have been due to social engineering. That is, the victims are normally tricked into executing them when they receive these malware as an email attachment or via messenger. In a targeted attack, the attacker would typically construct an email with a message specifically tailored to the recipient, and this email would contain either a file attachment or a link to a website with a drive-by-download attack.

Because of this increasing trend, there have been a lot of studies performed on targeted attacks. More specifically, Advanced Persistent Threats that involve Windows PCs, which is a subset of targeted attacks, are closely being analyzed and researched by many different groups. The reason is because Windows PCs are mainly used as workstations for systems of public infrastructure that tend to be attractive targets of cyber espionage or cyber sabotage.

1.3.1 Stuxnet: an Example of a High-Profile Attack

Stuxnet is a very well-known advanced persistent threat, a type of targeted attack, that started in 2005 based on information extracted from the registration information of the Command and Control (C&C) servers. It was first discovered and thoroughly analyzed by Symantec in July 2010. It is a very sophisticated malware that exploits 7 different vulnerabilities to infect other machines or terminals. One of the most important among the 7 vulnerabilities is "Microsoft Windows Shortcut LNK/PIF Files Automatic File Execution Vulnerability (CVE-2010-2568)". This was exploited mainly for the purpose of being able to penetrate into an air-gapped network via USB drives being used by engineers on the regular network and the air-gapped network. Upon penetration into the air-gapped network, the damaging payload of creating malfunction in a highly specialized operation is only executed under very specific configurations of the control systems. [Wue14]

This example simply exhibits how wide-scale and important targeted attacks have become, thus further emphasizing the importance of analyzing the next generation targeted attacks.

1.4 Stages of a Targeted Attack

Each targeted attack can be divided into multiple stages for easier analysis. There is no official standard for defining these stages. Each security vendor tends to have their own perception on how a targeted attack should be divided. It is still important to see how each stage is defined because it will give a better overall perspective on a targeted attack. Here are a few examples of breakdowns from well-known security vendors:

1.4.1 Stages of a Targeted Attack by Symantec

According to Symantec, a targeted attack can be divided into the following four stages [Sym14]:

1. **Incursion.** Penetrate into the target's network by either using social engineering tactics or exploiting vulnerabilities.
2. **Discovery.** Gain more information about the network topology of the company by exploring inside the network.
3. **Capture.** Take full control of the entire system.
4. **Exfiltration.** Steal the data.

1.4.2 Stages of a Targeted Attack by Trend Micro

Trend Micro interprets a targeted attack in the following sequences [Tre12]:

1. **Intelligence Gathering.** Use social networking sites to obtain publicly available information about certain individuals.
2. **Point of Entry.** Use social engineering or a website exploit to do the initial penetration into the individual's PC.
3. **Command and Control communication.** Method used by the hacker to control and navigate around the compromised network.

4. **Lateral Movement.** Invade into other machines in the corporation to gather more data.

5. **Asset/Data Discovery.** Use several techniques to determine where valuable sensitive data is stored at.

6. **Data Exfiltration.** Data to be stolen is compressed, encrypted and chunked before it is exported out of the network.

1.4.3 Stages of a Targeted Attack for this Paper

For the purposes of our research, we will define the stages of a targeted attack by the following:

1. **Invasion.** Malware attempts to enter into a network by penetrating into one of the terminals, either by exploiting a security vulnerability or by means of social engineering to fool a user to install it.

2. **Control.** Malware allows the hacker to take full control of the device by providing some form of a backdoor communications to accept commands from the hacker. In more sophisticated case studies, malware would also try to spread copies of itself via known virus propagation techniques to provide control of other machines on the network as well.

3. **Information Gathering.** Malware will now do some data mining on all possible terminals to look for data that can be of value to the hacker. Normally, malware would collect and dump these data into encrypted data files.

4. **Exfiltration.** Malware attempts to export these encrypted data files out of the network. To avoid being easily noticed by firewalls, malware would put it in a non-suspicious encrypted format, such as password-protected .rar files, and use standard ports and protocols such as HTTP POST to move the data out of the infected terminal into the C&C server.

1.5 Evidence of Targeted Attacks Shifting to Android

One of the most interesting literature found regarding usage of Android malware as part of targeted attacks is a research paper written by Trend Micro in 2012. While Trend Micro was investigating C&C servers involved in the Luckycat attack campaign, which were used to control infected Windows platforms, they also discovered Android files that appear to be proof-of-concepts of the Android component of a targeted attack, thus clearly demonstrating that it is still in early stages of development and have not yet been completed. The report also provides a disassembly of the Android artifacts found, which indicate that the hacker were trying to implement functionality to allow uploading and download of files to and from the infected Android device. However, the analysis is purely based on forensic analysis, and there are no clear indications of being able to reproduce an environment to simulate a live targeted attack on Android. [Tre12]

1.6 Android Permissions Scheme

Whenever a user installs an app, Android prompts a list of permissions that the app has requested. The main purpose is to allow the user to evaluate the security risk before installing the app based on the permissions that it has requested. However, it has been shown on a paper that out of the 150 most-installed apps, 58% of them require the INTERNET permission and 33% of them require the WRITE_EXTERNAL_STORAGE permission. According to the table of "dangerous" permissions that was presented, both of these permissions were ranked in the top 5 permissions requested by the 150 most-installed apps. Thus, since we would only need these 2 permissions to steal files and even maliciously edit data on the victim's SD Card, it would be relatively easy to use social engineering to fool someone into installing our app. [Jet11]

1.7 Arranging Files on Desktops

Since we will be analyzing files and directory structures on the Android device, we will briefly discuss research that has been done on how people tend to organize files on Windows machines. According to the dissertation of Henderson, an interview has been conducted to 10 people on how each person organizes files on their PCs. Based on the results, the trend that we can notice is that people tend to create a hierarchy of folders to describe the context of each file. A few examples provided on one of Henderson's figures to demonstrate this concept are: *"Labs\2004\INFOSYS 221", "Labs\2004\INFOSYS 222", "Labs\2005\INFOSYS 221"*

However, most of the files that matter to them are found in either the Desktop or My Documents. Another common trait that we have noticed is that the more important or frequently accessed a document is, the greater the likelihood that the file is located closer to the top level folder. One common practice among some of the participants is that as time passes by, house cleaning of files is postponed and therefore most of the files tend to be placed in a relatively flat structure on the Desktop. [Hen09]

1.8 Problem Statement

Most of the technical research involved in cyber security for targeted attacks tend to focus more on Windows platform, since Windows is widely used in corporate environments and IT systems of major infrastructures. However, Android should not be disregarded since it has emerged as the most widely used platform for smartphones. There have been many research conducted on Android malware families, as well as malware sandboxes that have been created for Android, but not really for the overall study of a targeted attack. Lately however, there have been reports about how Android devices are now being pursued for targeted attacks, but none of these reports provide a clear demonstration of how it is possible. For this paper, I will focus my attention to how an environment simulated for a known case study of a Windows targeted attack can be modified to accommodate an Android device instead. The changes required are not trivial, but it is not impossible to implement. By being able to prove that Android devices can be subject to targeted attacks and clearly demonstrating it on a test bed, we can use the same test bed to build future products based on protecting Android devices against targeted attacks. Afterwards, we will define the constraints on the Android device, and do a scientific approach on trying to work around these constraints to extract data that can yield useful information.

This research will attempt to answer the following questions:

1. How feasible is it to retarget an attack from Windows to Android?
2. For the control phase, what type of payloads can the attacker deliver from the C&C server to the infected Android device?
3. For the information gathering phase, based on the resource constraints discovered and performing a scientific analysis, what is an effective way of mining information out of an Android device? Is it clustering filenames or partially previewing certain documents?

1.9 Practical Applications

This research can have good applications in the real world because of many reasons. Until then, performing a targeted attack on Android devices has merely been a proof of concept, as demonstrated by the LuckyCat attack campaign. We want to prove that it is possible, and that this will become a widespread problem. Also, by being able to build a virtual environment that allows a full simulation of a cyber attack, it can be used as a testbed for comparative analysis of existing security products on Android. In fact, most antivirus vendors require their engineers to be able to reproduce an attack in-house so that they can obtain forensic data such as network packet captures during penetration or files and other residues left behind from the attack. This is the most reliable way to test whether a security product can perform up to its expectations, which is to detect and stop an attack. Another point is that by understanding which vectors of attack are feasible and not feasible, a security professional can focus on the correct areas and build the defense mechanisms based on that information.

This research also conducted a scientific investigation on previewing PDF files on Android. While we were analyzing and verifying the investigation results, we discovered that the previewed data has been effective in predicting the intentions or planned activities of the owner of the device. In theory, this concept can be used by law enforcement in predicting the intentions of criminals who have android smartphones.

CHAPTER 2

Design and Implementation

We propose the following procedures in undertaking the research. We will create a C&C server that can control a Windows terminal infected with Greencat, a Windows malware part of a targeted attack, and test the communications protocol. Then we will create an Android app that uses the same communications protocol to be able to receive commands from our C&C server. After analyzing the results, we would define what the constraints are on an Android environment that would affect the attacker's data theft intention. Thus, based on this assumption, we would then come up with an optimal way to process files and directories, and determine the best way to extract the maximum value of information from the file system. We would then realize that previewing PDF files is an effective way of obtaining information about the victim, so we would then come up with a proposal on how to effectively analyze PDF files.

2.1 Development Tools

In creating our C&C server, we have decided to use ruby and sinatra to write the code due to multiple reasons. First of all, ruby and sinatra allows us to write very concise code, since the underlying steps to do HTTP requests is already implemented. It allows us to spend more time on defining the communication protocol used by Greencat rather than trying to implement the detailed steps of an HTTP communication. On the C&C server, we will use an sqlite database to store all of the logs and files that have been taken from the victim machines. Sqlite is a better choice than MySQL because we don't need to do intensive database operations, and its single flat-file structure makes it easier to manage the files. In addition, one of the main reasons that these technologies were selected is because of their feasibility to be installed and used on a standard Amazon EC2 Instance (Amazon's cloud hosting service).

For our Android apps, we will use Android SDK 2.3.3 because this is the lowest SDK version that has all of the features we need for our prototypes. Here is a list of 3rd party libraries that we will be using and the corresponding reasons.

JCIFS ver. 1.3.17 [JCI11] - Open source library that allows an Android app to access SMB shares in Windows. This will be used so that the Android app can access an open Windows share and copy a file from it.

j256 simplemagic ver 1.6 [Wat14] - A library that allows an Android app to determine the file type based on file content data. This will allow effective filtering of files during SD Card analysis.

iText PDF 5.4.5 [Ite14] - An open source library that allows an Android app to extract text from PDF files.

2.2 Understanding Greencat Communications Protocol

A good thorough understanding of the Greencat's communications protocol is required. We downloaded and reverse engineered a Greencat malware sample. Based on our analysis, Greencat uses HTTP GET and POST to do all of its communications with the C&C server. Here are the detailed steps of Greencat's communications protocol.

1. Upon execution, Greencat will send an HTTP GET request to the hacker's C&C server to inform the attacker that it has successfully infected a Windows machine and it is now ready to accept commands. The exact path of

Operating System	Version
Windows XP	5.1
Windows XP 64-bit	5.2
Windows Server 2003	5.2
Windows Vista	6.0
Windows 7	6.1
Windows 8	6.2
Windows 8.1	6.3

Table 2.1: Important OS Version Numbers by MSDN [MSD14].

the request is */worlda.html*

Due to the modified hosts file, this traffic has been automatically redirected to our C&C server. The User-Agent field in the HTTP header of this GET request will be in the following format:

User-Agent: <majorver>.<minorver> <hr>:<min> <comp>\<user>

Here is an example:

User-Agent: 5.1 21:44 LATTE\joedoe

Definitions of the tokens:

<majorver>.<minorver> is the version number of Windows OS. Please refer to Table 2.1 for a list of version numbers for the important OS.

<hr>:<min> is the local time on the infected machine left-padded with zeros (e.g. *06:34*).

<comp> is the Windows computer name.

<user> is the Windows user currently logged in.

2. The C&C server replies with an HTTP response that uses the following format in its response body:

   <!–<img border=1 src="/*<enc-url>*/#*<enc-compname>*.opt">–>

   where *<enc-url>* is the encrypted URL for the C&C server, and *<enc-*

enc	dec	enc	dec	enc	dec	enc	dec
a	4	r	h	I	y	Z	P
b	5	s	i	J	z	0	Q
c	6	t	j	K	A	1	R
d	7	u	k	L	B	2	S
e	8	v	l	M	C	3	T
f	9	w	m	N	D	4	U
g	.	x	n	O	E	5	V
h	_	y	o	P	F	6	W
i	/	z	p	Q	G	7	X
j	-	A	q	R	H	8	Y
k	a	B	r	S	I	9	Z
l	b	C	s	T	J	.	0
m	c	D	t	U	K	_	1
n	d	E	u	V	L	/	2
o	e	F	v	W	M	-	3
p	f	G	w	X	N		
q	g	H	x	Y	O		

Table 2.2: Decryption table to decipher the response from the C&C server.

compname> is the encrypted computer name of the targeted PC. To decrypt the URL and the computer name, we use the decryption table displayed at Table 2.2. To decipher, simply look for the character at column <*enc*>, and then replace it with the corresponding <*dec*> character.

An example would be:

<!–<img border=1 src="/*rDDz#ii_.g.g/g/imwn/#kxI*.dummy">–>

Here, the conversion would be as follows:

r → h

D → t

D → t

z → p

and so on…

The resulting decrypted command would be:

<!–<img border=1 src="/*http://10.0.2.2/cmd/#any*.dummy">–>

3. The Windows victim will then extract the decrypted URL, and send an HTTP GET request to it. In this case, the decrypted URL is:
   http://10.0.2.2/cmd
   Now it has notified the C&C server that it is ready to accept commands.

4. The C&C server will search its own database for a list of commands inputted by the attacker with the matching host name of the victim. If a command is found, the C&C server sends an HTTP response with the command embedded in its response body. Based on our malware analysis, here is a comprehensive list of all the possible commands for this Greencat sample and its description.

   (a) *shell*
   launch a command shell to execute DOS commands

   (b) *list </p | /s | /d>*
   list processes, services or directories

   (c) *kill </p | /s> <PID | service name>*
   kill a process or a service

   (d) *getf <filename> dummy dummy <url>*
   After receiving this command from the C&C server, the victim machine will read the data of <filename>, and send an HTTP GET request to <url> with the file data contained in the request body. <url> normally points to a script on the C&C server that would specifically retrieve the data content from its local database. After completing the entire file transfer, the victim machine will send an HTTP GET request with the symbol $ as the only character in the request body.

   (e) *putf <filename> dummy <file size> <url>*
   The C&C server is instructing the victim machine to download from <url> and save it locally with the filename <filename>.

(f) *start </p | /s> <program name | service name>*

Launches a program or starts a service.

(g) *whoami*

Get the user name.

(h) *quit*

The server is done with its commands.

(i) *v*

Get the version number of the malware.

(j) *pidrun <pid> <filename>*

Execute the filename with the same privileges as process with pid number.

(k) *geturl <URL> <filename>*

Victim should visit the URL and save its contents.

5. If the hacker sends a command that is not included in the list above, Greencat will assume that it has to be executed thru the Command Prompt. It will either prompt the user to launch Command Prompt first via shell command if it has not yet done so, or it will send the command to the Command Prompt process for execution.

6. Greencat takes the console output from the command that was just executed, and then sends an HTTP GET request to the C&C server with this output embedded in its request body.

   Steps 4, 5 and 6 will be repeated until the victim machine receives the "quit" command from the attacker, which is an indication that there are no commands left to be processed by this victim machine.

2.3 Setting up the Windows Machine for Infection

Since we will be using a live malware sample on the Windows machine, it is important that the machine can be easily restored back to pre-infection settings. Thus, we have decided to use a Vmware virtual machine. It has been setup with the following specifications:

1. OS: Windows XP SP2
2. Network IP address: 172.16.102.180
3. Local hosts file *C:\WINDOWS\System32\Drivers\etc\hosts* includes an additional entry that assigns our IP address to the hostname of the original C&C server.
4. Max. Transmit rate to router: 145 Mbps
5. Max. DSL speed of router: 50 Mbps D / 10 Mbps U

2.4 Creating the C&C Server

We created a C&C server based on the Greencat communications protocol elaborated in the earlier section. The main objective of the C&C server is to be able to control both infected Windows and Android devices.

2.4.1 Implementation of C&C Server

The C&C Server listens on port 80 and 443 using standard HTTP protocol, since Greencat communications uses HTTP protocol for its transactions. Here is a comprehensive list of HTTP GET requests that it will accept and process.

Here are the specifications of the C&C server.

1. Ability to send commands to any victim device based on device name

2. Ability to host a file and instruct any victim to download it
3. Ability to get a file from any victim and save it on its database
4. Listens on both port 80 and 443 as required by this Greencat sample.
5. Must be able to run on an Amazon EC2 Instance.

2.4.2 Requests to C&C Server

Here is a quick summary of what GET requests the C&C Server will accept. Note that its details have already been explained earlier in Section 3.1 Understanding Greencat's Communications Protocol.

/worlda.html

/cmd/

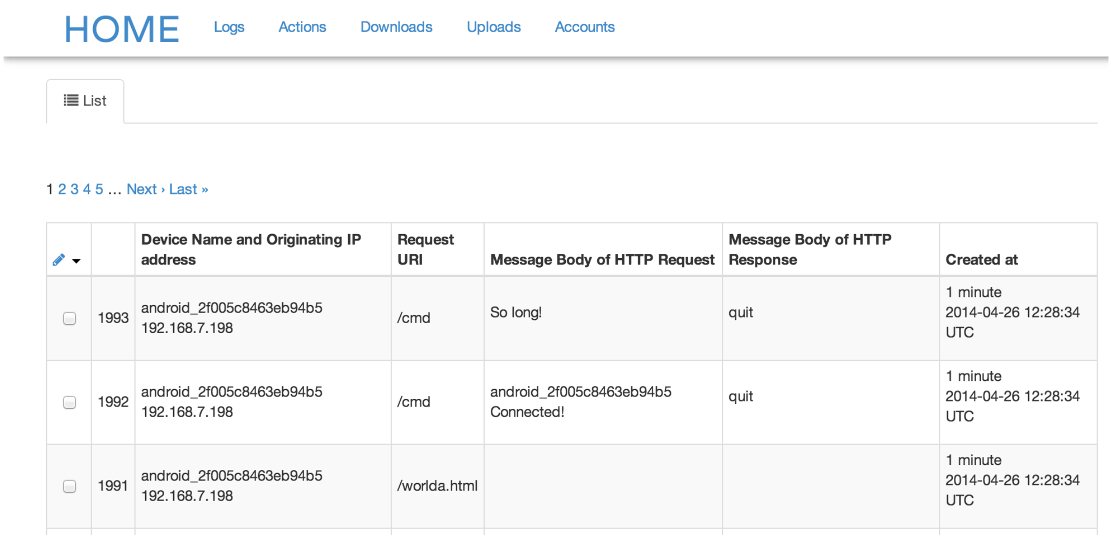
/getf/

/putf/<id>

2.4.3 User Interface of the C&C Server

A screenshot of the top page of the user interface is shown in Figure 2.1. Here is a quick explanation of what each menu item does.

1. Logs - view each request received and responses made by the C&C server.
2. Actions - Send a command to a specific device name.
3. Downloads - Host any file on the C&C server and push it to any device that it controls.
4. Uploads - List of files stolen from all infected terminals. This will also be used as the repository for the PDF analysis that will be created by the



		Device Name and Originating IP address	Request URI	Message Body of HTTP Request	Message Body of HTTP Response	Created at
☐	1993	android_2f005c8463eb94b5 192.168.7.198	/cmd	So long!	quit	1 minute 2014-04-26 12:28:34 UTC
☐	1992	android_2f005c8463eb94b5 192.168.7.198	/cmd	android_2f005c8463eb94b5 Connected!	quit	1 minute 2014-04-26 12:28:34 UTC
☐	1991	android_2f005c8463eb94b5 192.168.7.198	/worlda.html			1 minute 2014-04-26 12:28:34 UTC

Figure 2.1: Administration Panel for C&C Server

Android app later.

5. Accounts - user accounts for the control center

2.4.4 Testing the C&C Server

After creating the C&C server, we tested a comprehensive list of commands that Greencat malware supports by design. Table 2.3 shows a list of the commands that were sent from the C&C server to the Windows victim machine and the corresponding results of each command.

Command Sent	Test Result on Windows
initial request	Passed
shell	Passed
a shell command e.g. *dir c:.**	Passed
list /p	Passed
list /s	Passed
list /d	Passed
kill /p	Passed
kill /s	Passed
getf	Passed
putf	Passed
start /p	Passed
start /s	Passed
whoami	Passed
quit	Passed
v	Passed
geturl	Passed

Table 2.3: List of Greencat commands tested on Windows.

CHAPTER 3

Android Attack

3.1 Retargeting the Attack to Android

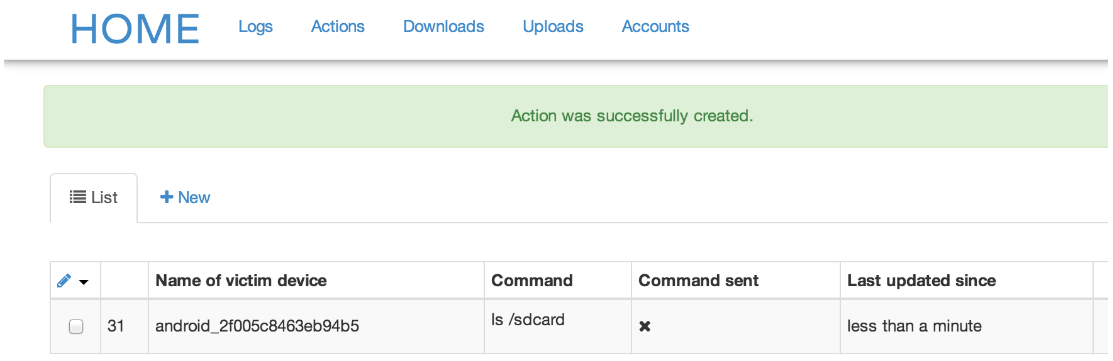
As shown in the existing literature, the Luckycat campaign's objective in their attempts to build an Android targeted attack is to upload and download files, thus showing the intent of data theft. Thus, this research will focus on the data theft aspect of a targeted attack for Android. We have created a new Android app from ground up that can communicate with the C&C server. This requirement is important because an attacker would prefer to use one C&C server to control all devices regardless of its operating system. Some of the commands that have been ported over from Windows to Android are more focused on data theft. Also, since the assumption is that the Android device is in its default factory setting as much as possible, meaning that it is not rooted, the attacker would have limited privileges to what he can do, and yet it will be demonstrated that the attacker is able to accomplish so much more as opposed to attacking a Windows device.

A detailed set of sequential test cases will now be elaborated to demonstrate that the prototype created works properly. In the Windows version, Greencat uses https (port 443) to do the file transfers to avoid eavesdroppers from being able to see the traffic. However, for this research, we will simplify the implementation and perform all of the operations via HTTP (port 80) only.

1. **Install the app and report to C&C server.** For this 1st test case, we installed the Android app to an Android smartphone. It successfully

connected to the C&C server and reported its network hostname and IP address. There are no commands to execute, so it quits immediately after connecting, as it is seen on the logs.

2. **Generate a directory listing.** The 2nd test case is to do a directory listing on the SD Card, to see if the attacker can find any interesting files. Thus, we issue the ls /sdcard command as shown in Figure 3.1.



Action was successfully created.

		Name of victim device	Command	Command sent	Last updated since
☐	31	android_2f005c8463eb94b5	ls /sdcard	✖	less than a minute

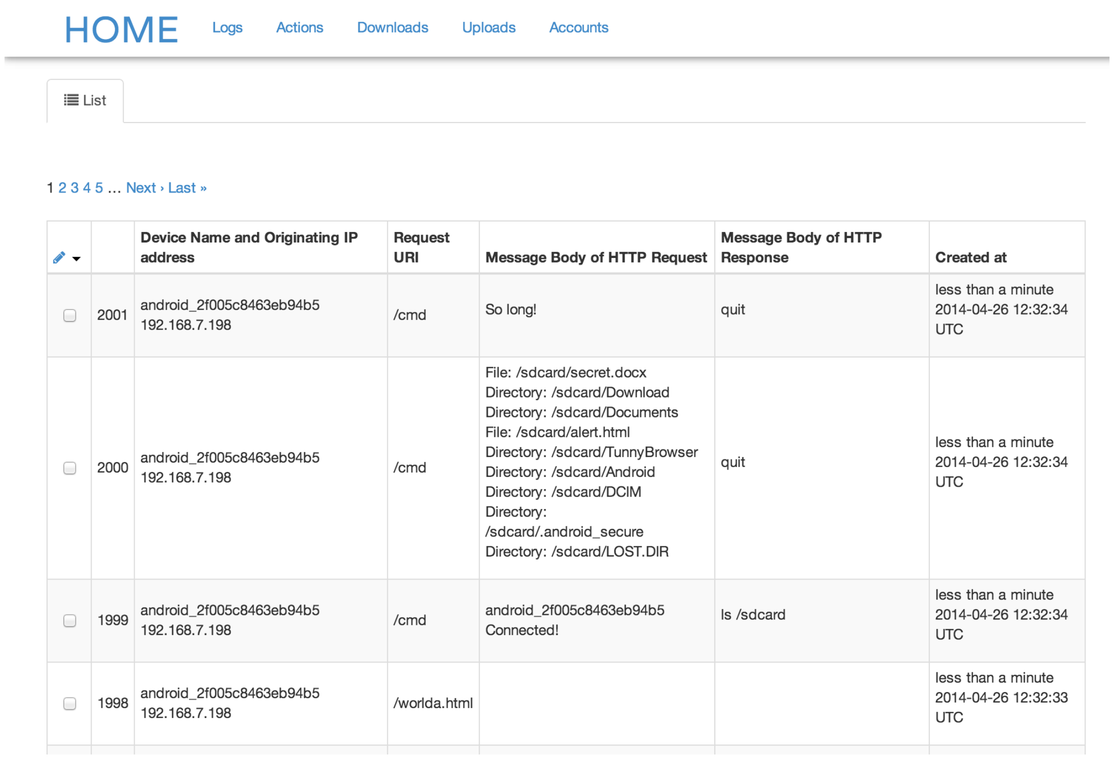
Figure 3.1: Issuing the directory listing command.

The C&C server receives the response as shown in Figure 3.2. Based on the results of the directory listing, it appears that the SD Card contains an interesting file named *secret.docx*.

3. **Steal a file.** In the 3rd test case, the attacker would like to retrieve a copy of *secret.docx* that was found in the 2nd test case, and store the document in the C&C server. Thus, a command to steal the file has been issued as shown in Figure 3.3. The results of the 3rd test case can be seen in Figure 3.4. As exhibited in the figure, the infected terminal reports to the C&C server, indicating that it is ready to accept commands. The C&C server then sends the following command:

getf /sdcard/secret.docx dummy dummy http://192.168.7.149/getf

This is instructing the infected device to load the binary content of /sd-

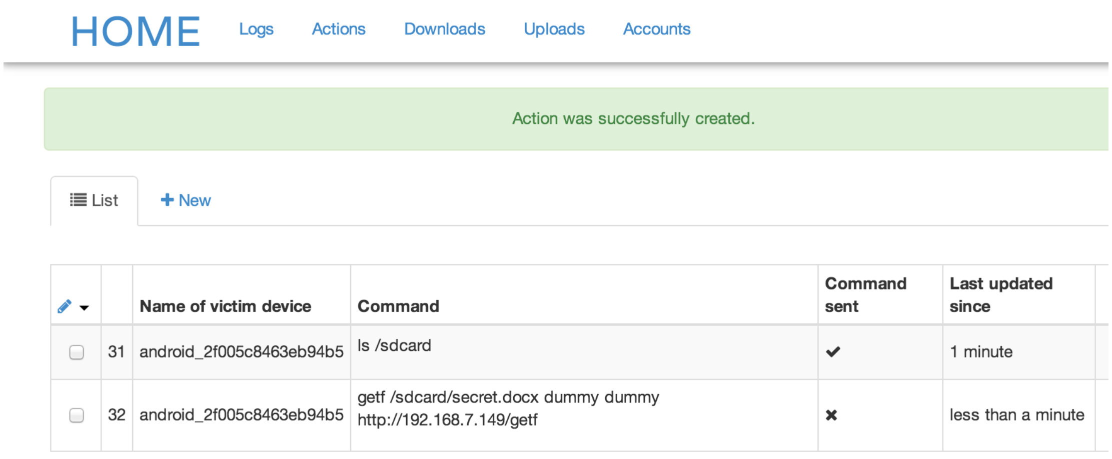


		Device Name and Originating IP address	Request URI	Message Body of HTTP Request	Message Body of HTTP Response	Created at
☐	2001	android_2f005c8463eb94b5 192.168.7.198	/cmd	So long!	quit	less than a minute 2014-04-26 12:32:34 UTC
☐	2000	android_2f005c8463eb94b5 192.168.7.198	/cmd	File: /sdcard/secret.docx Directory: /sdcard/Download Directory: /sdcard/Documents File: /sdcard/alert.html Directory: /sdcard/TunnyBrowser Directory: /sdcard/Android Directory: /sdcard/DCIM Directory: /sdcard/.android_secure Directory: /sdcard/LOST.DIR	quit	less than a minute 2014-04-26 12:32:34 UTC
☐	1999	android_2f005c8463eb94b5 192.168.7.198	/cmd	android_2f005c8463eb94b5 Connected!	ls /sdcard	less than a minute 2014-04-26 12:32:34 UTC
☐	1998	android_2f005c8463eb94b5 192.168.7.198	/worlda.html			less than a minute 2014-04-26 12:32:33 UTC

Figure 3.2: Results of directory listing command. Please note that the records are displayed in reverse chronological sequence as indicated by their row ID.

card/secret.docx into an HTTP POST request body, and send that request to *http://192.168.7.149/getf* . The C&C server will then store the binary content into the database. After all of the binary content has been sent, the infected device will send a $ symbol, indicating that the transfer is complete. This is the communications protocol used in Greencat when the C&C server wants to retrieve a file from any infected terminal.

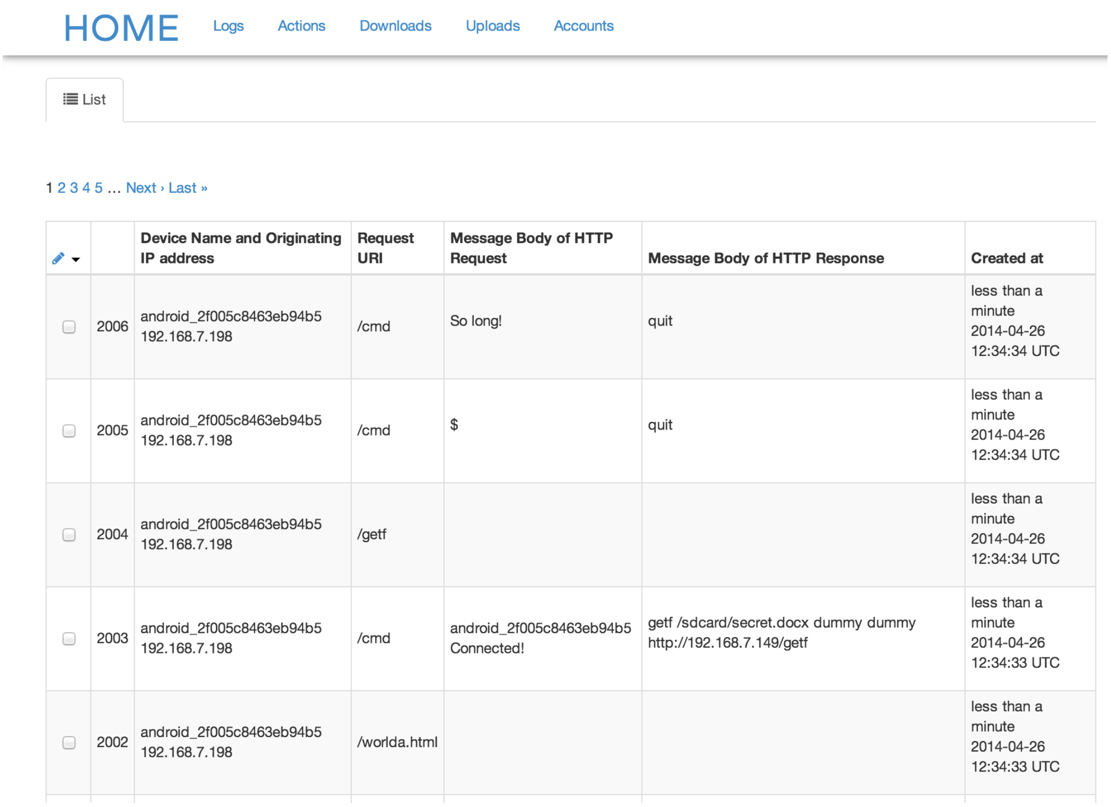
Clicking on the Uploads menu item will display a list similar to Figure 3.5. It shows an entry referring to a copy of the file */sdcard/secret.docx* stored in the database of the C&C server. Thus, the 3rd test case is successful.



Action was successfully created.

		Name of victim device	Command	Command sent	Last updated since
☐	31	android_2f005c8463eb94b5	ls /sdcard	✓	1 minute
☐	32	android_2f005c8463eb94b5	getf /sdcard/secret.docx dummy dummy http://192.168.7.149/getf	✖	less than a minute

Figure 3.3: Issuing a command to steal a file.

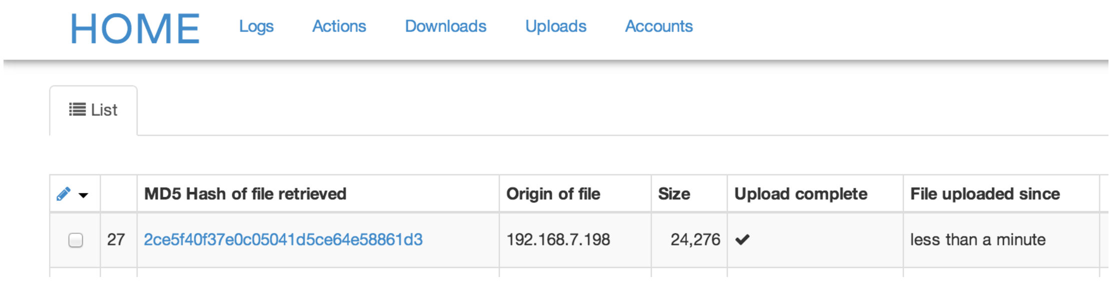


		Device Name and Originating IP address	Request URI	Message Body of HTTP Request	Message Body of HTTP Response	Created at
☐	2006	android_2f005c8463eb94b5 192.168.7.198	/cmd	So long!	quit	less than a minute 2014-04-26 12:34:34 UTC
☐	2005	android_2f005c8463eb94b5 192.168.7.198	/cmd	$	quit	less than a minute 2014-04-26 12:34:34 UTC
☐	2004	android_2f005c8463eb94b5 192.168.7.198	/getf			less than a minute 2014-04-26 12:34:34 UTC
☐	2003	android_2f005c8463eb94b5 192.168.7.198	/cmd	android_2f005c8463eb94b5 Connected!	getf /sdcard/secret.docx dummy dummy http://192.168.7.149/getf	less than a minute 2014-04-26 12:34:33 UTC
☐	2002	android_2f005c8463eb94b5 192.168.7.198	/worlda.html			less than a minute 2014-04-26 12:34:33 UTC

Figure 3.4: Results of command to steal a file.

3.2 Payload Retargeted to Android

We decided to take this a step further and implement commands that are specific to the Android device to demonstrate why some types of payloads on Android


HOME Logs Actions Downloads Uploads Accounts

List

		MD5 Hash of file retrieved	Origin of file	Size	Upload complete	File uploaded since
☐	27	2ce5f40f37e0c05041d5ce64e58861d3	192.168.7.198	24,276	✓	less than a minute

Figure 3.5: List of files stolen from victims.

are more dangerous than of Windows. Android's permission system may allow a developer to access SMS messages, addressbook contact information and call logs on a device as long as these are requested during installation [Go14d] . Here is a comprehensive list of commands that have been implemented that is considered unique to Android. We are also including the samples of logs to prove the success of our experiments:

1. **readsms**. Instructs the infected device to put all the SMS messages in the request body. The format used is as follows:

   From: <phone number> : <SMS body>

2. **readcontacts**. Instructs the infected device to enumerate all of the contacts in the phone addressbook and the SIM card, and put them all in the request body. The format is as follows:

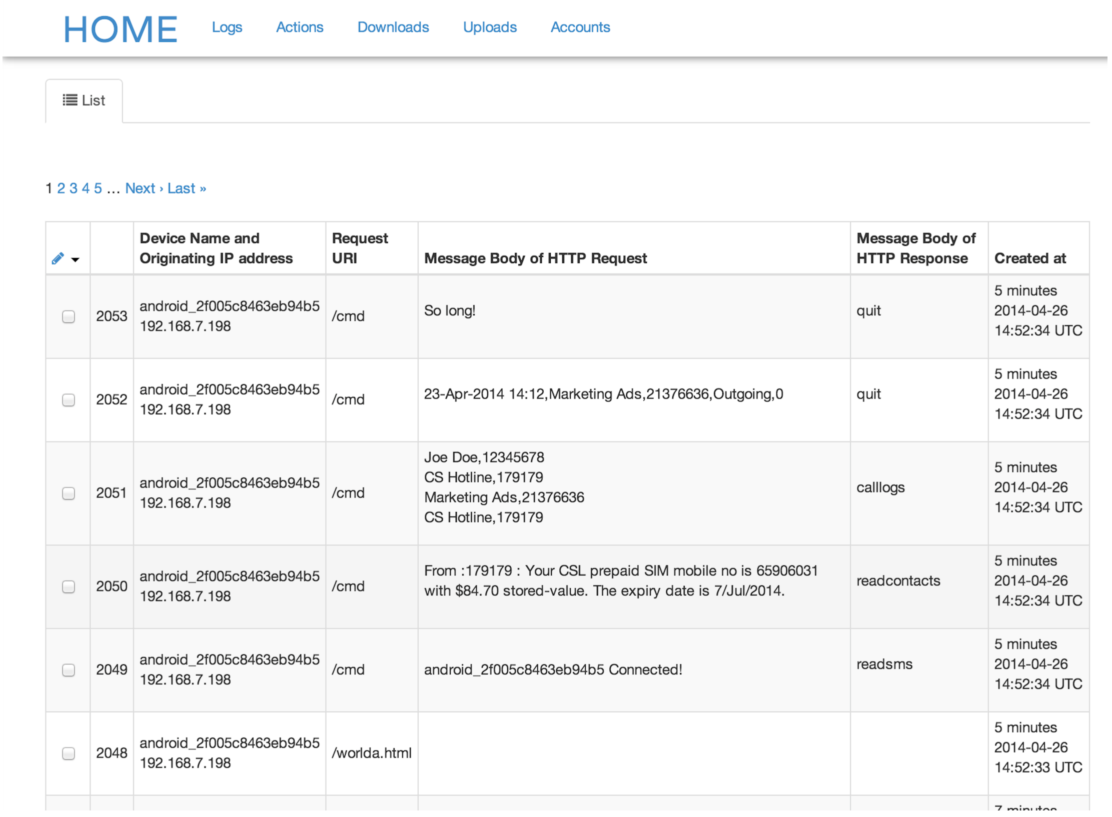
   <name>, <phone number>

3. **calllogs**. Instructs the infected device to put all of the incoming and outgoing calls made in the request body. The format is as follows:

   <date time>,<name>,<number>,[Incoming|Outgoing]*,<duration>*

   Please refer to Figure 3.6 to see the success of all 3 instructions above.

4. **stealdocs.** A feature has been implemented in the Android app that will automatically steal specific files that it finds in any open Windows shared



		Device Name and Originating IP address	Request URI	Message Body of HTTP Request	Message Body of HTTP Response	Created at
	2053	android_2f005c8463eb94b5 192.168.7.198	/cmd	So long!	quit	5 minutes 2014-04-26 14:52:34 UTC
	2052	android_2f005c8463eb94b5 192.168.7.198	/cmd	23-Apr-2014 14:12,Marketing Ads,21376636,Outgoing,0	quit	5 minutes 2014-04-26 14:52:34 UTC
	2051	android_2f005c8463eb94b5 192.168.7.198	/cmd	Joe Doe,12345678 CS Hotline,179179 Marketing Ads,21376636 CS Hotline,179179	calllogs	5 minutes 2014-04-26 14:52:34 UTC
	2050	android_2f005c8463eb94b5 192.168.7.198	/cmd	From :179179 : Your CSL prepaid SIM mobile no is 65906031 with $84.70 stored-value. The expiry date is 7/Jul/2014.	readcontacts	5 minutes 2014-04-26 14:52:34 UTC
	2049	android_2f005c8463eb94b5 192.168.7.198	/cmd	android_2f005c8463eb94b5 Connected!	readsms	5 minutes 2014-04-26 14:52:34 UTC
	2048	android_2f005c8463eb94b5 192.168.7.198	/worlda.html			5 minutes 2014-04-26 14:52:33 UTC

Figure 3.6: Results for readsms, readcontacts and calllogs.

networks. This command instructs the Android app to scan any open SMB ports (TCP 139) on the network to search for any open Windows shared drives. It takes the IP address of eth0 or eth1, which is most likely the network interface connected to the internal network, and does a port scan of TCP 139 thru the entire subnet of 255.255.255.0. However, if port 139 is blocked on the machine by the firewall, there will be no response to the port scan ping. Thus, to take this common situation into account, a timeout of 200 ms has been implemented to avoid lengthy waits on requests that firewalls drop. Once an open SMB port is found, the Android app attempts to access it via default guest privileges. If the share is accessible, the app will traverse through the 1st level directory to search for any files whose names contain the string ".doc". Please note that the app can be easily modified to do a recursive search for a few more levels of sub-directories, but it has been configured to scan only the 1st level for the purposes of this experiment. Test runs have been conducted by placing multiple files of various sizes with ".doc" extension on the open share folder.

The network diagram of this setup to illustrate the expected speeds of network transmissions is displayed at Figure 3.7.

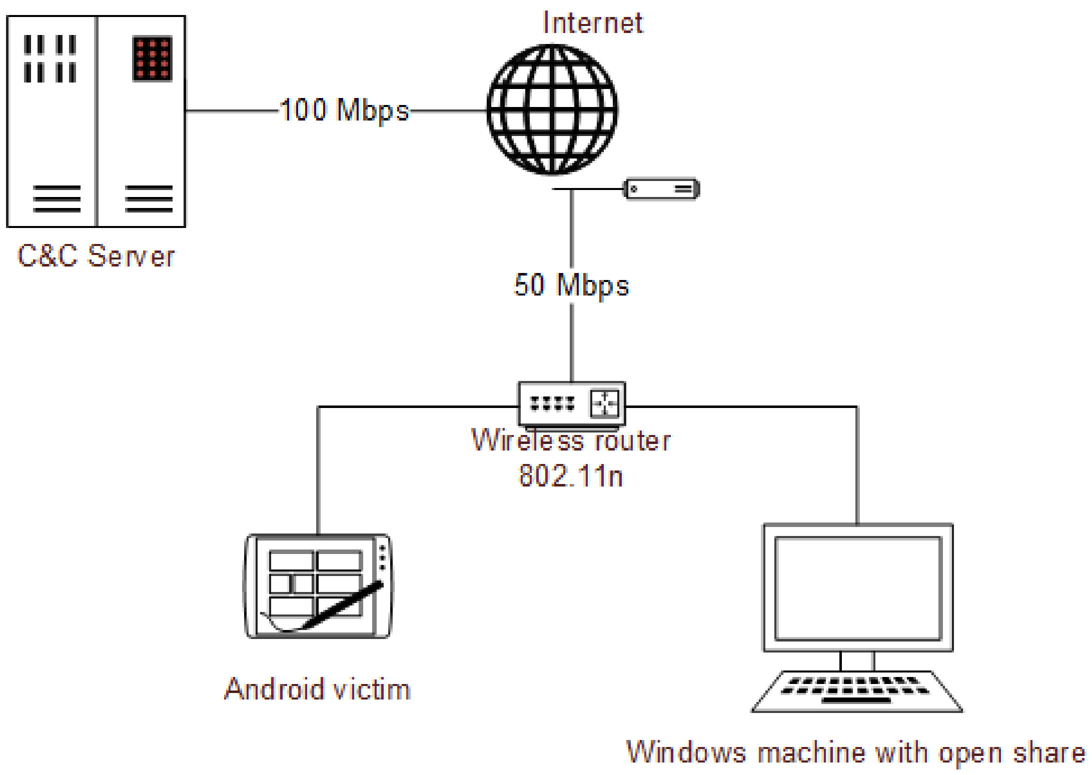


Figure 3.7: Network diagram of test environment for traversing open shares from Android to Windows.

The benchmarking results is listed at Table 3.1.

Based on the results, the average speed to steal a file (including latency and propagation time) is about 40kb/s, thus exhibiting that it is feasible to steal only small files, and significant improvements need to be made so that more data can be exfiltrated from the device. However, it needs to be emphasized that for this test scenario, the smartphone is connected to the router, which is connected to the Internet via DSL line. Most of the time, smartphones

ID	File size	Total time to steal	Avg. speed	Result
A	24 kb	1.5 sec	16 kb/s	Passed
B	148 kb	3.5 sec	42 kb/s	Passed
C	560 kb	14.3 sec	39 kb/s	Passed
D	1,007 kb	26.3 sec	38 kb/s	Passed
E	8,786 kb	3 min 36.9 sec	41 kb/s	Passed
F	81,524 kb	Did not finish	N/A	Failed

Table 3.1: Test results for uploading files from open Windows shares to C&C server via Android smartphone.

will only be connected via 3G data when user is on the road. Thus, it can be concluded that it is best to keep the data transfer amount to a minimum. In the next chapter, we will demonstrate scientific approaches in determining the most optimal way to select the type of data to transfer for maximum ROI in terms of information.

3.3 Other Payload Implemented on Android

Another feature that was implemented on the Android app is registering a ContentObserver object [Go14b] for Browser.BOOKMARKS_URI [Go14c] events to allow observing the browser events on the default Android browser. Basically, after a user surfs to a web page each time, it will trigger an event handler in the Android app, and allow it to retrieve the URL and the page content. This would allow the Android app to do some inspection to see if the user is doing something illegal, for example, surfing child pornography sites. Potentially, the Android app can report this incident to the C&C server and display a warning alert to the Android user to discourage him from continuing with the illegal activity.

Another useful feature implemented on the Android app is a FileObserver to the SD Card, which allows monitoring of file write events [Go14a]. By observing the SD Card for any file write events, the app can report any new files added in the SD Card to the C&C server. If the filename appear to be of great interest, for example, a file entitled "secret blueprints to the assembly line.docx", then the app can report this to the C&C server to notify the attacker, and then the attacker can instruct the app to upload the file to its C&C server.

There are other features that can be implemented which could make this app more dangerous and useful to the attacker. These features will be listed at Section 5.2.

3.4 Bypassing Resource Constraints on Android

As we have seen on the test results from stealing a file, one of the major constraints on an Android device is the network speed. Smartphones tend to have less powerful hardware compared to ordinary desktops or laptops. In addition, the file test above was conducted with the Android device connected to a DSL line. In reality, an Android device will have the following typical situations.

1. WiFi will never stay on all the time since it disconnects automatically in stand-by mode on most of the phones' default settings.

2. 3G mobile data might be always on, but it will be relatively slower especially its upload speed.

Thus, we need to come up with the best approach on how to extract the most information with such restrictions. The next chapter shall discuss two methods that we will explore, and see if at least one of them is effective.

CHAPTER 4

Data Analysis in the Dark on Android

This chapter will demonstrate our attempts to use two different concepts in trying to extract information from the Android device while trying to minimize data transfers due to networking limitations as shown earlier. The first concept is creating clusters of filenames. The second concept is limited previewing of PDF files.

4.1 Clustering Based on Filenames

4.1.1 File and Directory Arrangement on Android

Henderson's dissertation talks about the different file arrangements on Windows, and how some users put best effort in doing so. However, on a Android smartphone, it is more unlikely for someone to organize his files based on folder hierarchy structure, because the small screen and the lack of a hard keyboard makes it more difficult to create the folders and move files around. However, there were a few traits mentioned that are very likely to be applicable on Android as well, such as the following:

1. Path names still tend to provide a good context about the file, since we discovered that the path names will most likely describe the app that created the files and what type of files they are.

2. Some of the participants in the survey had the tendency to let the files fall

on a flat structure when they are too busy to create well-structured folders to arrange them. This is more likely to be even truer on Android.

4.1.2 Simple Algorithm and Implementation of Clustering

As a way of getting information from the victim, we want to determine if it is feasible to create clusters of filenames, select a filename from each cluster and upload the corresponding file to the C&C server. If this algorithm works perfectly, the idea is to have any file represent all the other files within its cluster, and then we can simply retrieve one file from every cluster to minimize the data transfer. A new Android app has been specifically to do only this part of the research so that our testers would be more willing to participate in this experiment.

The filenames within the same cluster must meet the following criteria, which will be explained in detail.

1. **Same numerical file type classification.** We have defined our own numerical values for certain file types. This list is displayed in Table 4.1. Basically, we exclude all files from further analysis that have been classified with file type 0. The library simplemagic has been used to do the file type classification. It reads part of the actual file content to determine its file type. Reading the file content is actually better than using the file extension for file type determination because some apps tend to use ambiguous extensions when naming their files, such as using *.dat* to name some of their image files.

2. **Same filename-based score.** Assign a numerical weight for each character based on the following set of rules, and then add up all the weights to generate the filename-based score.

   (a) 5 points for 0 to 9, a to f, A to F.

(b) 3 points for g to z, G to Z.

(c) 1 point for all other characters (including white space and punctuation marks).

3. **Same pathname-based score.**

   (a) Change all numerical digits (0-9) to '5'. For example, if the pathname is *myfolder123*, then for calculation purposes we would assume that it is *myfolder555*.

   (b) Convert each character to their corresponding ASCII numeric value.

   (c) Sum up all the ASCII numeric values to get the pathname-based score. So theoretically, *tree/myfolder123* has the same score as *tree/myfolder789*.

File type	Description of files
0	No determination has been made, or file type is not of interest.
1	Images (PNG, JPEG, GIF)
2	Office documents (PPTX, XLSX, DOCX, PPT, XLS, DOC)
3	PDF documents
4	Video files (MP4A)

Table 4.1: Self-assigned numerical file types.

The approach will be as follows.

1. The app would traverse the entire SD Card and enumerate each file.

2. Use the library simplemagic to determine the file type. Only files with non-zero file type will be accepted for further analysis.

3. For every file accepted, generate the filename-based score and the pathname-based score.

This app has been executed on two different devices that are primarily used by their respective owners. Person A ran the app on his device. We processed the logs to count the nodes and the clusters, and a summary is shown in Table 4.2. Noticed that there are 46 single-node clusters, and there are 3 clusters with 65 or more nodes per cluster. Thus, we decided to investigate these two issues in detail by examining their raw logs. Further inspection of the 3 clusters leads to the details shown in Table 4.3.

Node(s) per cluster	Total cluster count
1	46
2	10
3	1
4	4
5	1
6	2
9	1
12	1
13	1
20	1
28	1
29	1
65	1
71	2

Table 4.2: Cluster count based on nodes per cluster for Person A.

Nodes per cluster	File type	Path and Filename pattern
71	images	/DCIM/Camera/<date>_<time>.jpg
71	images	/DCIM/.thumbnails/<13-digit number>.jpg
65	images	/Android/data/com.linkedin.android/cache/ li_images/<32-hex char>.0

Table 4.3: Details of clusters with many nodes for Person A.

The first row in Table 4.3 refers to all of the camera shots that have been taken by the smartphone. We looked at the logs which also include the image resolution, and it appears that most of them were taken in the default camera resolution, which is approximately 4.8 megapixels. However, we also discovered that some of these images are completely unrelated with each other, besides the fact that they were taken using the same smartphone. Thus, we would have to say that the clustering operation failed in this part of the scenario.

We also examined the raw logs for the 46 single-node clusters. One thing that caught our attention is that almost all of the PDF files in the device were in single-node clusters. This makes sense, since every PDF is most likely downloaded from different sources with different material content. However, this would be a problem, because some users may have hundreds of downloaded PDFs on their devices. It would be difficult to upload each PDF file from the single-node cluster to the C&C server due to network limitations. Thus, the clustering operation here did not work as well.

The app also ran on Person B's device. We compiled the raw logs, and Table 4.4 shows the summary of the clustering results. According to this data, we have a total of 5652 nodes in 442 clusters. However, similar to the earlier example, there are 303 single-node clusters which is too many. Also, the top 5 clusters already cover 4605 nodes, which is more than 81% of the total number of nodes. We took the raw logs for the top 5 clusters for further analysis, and the summary can be seen in Table 4.5.

It appears that all of these pathnames belong to an app called Dropbox. */emulated/0/cloudagent/thumbnail/* is where Dropbox stores the thumbnail images for processing. This is actually not considered very valuable, since an attacker would prefer to have the original high-resolution photos. However, the 2nd entry, which is *cloudagent/cache/Dropbox/Camera Uploads/<date and time>*, are actual camera shot images taken by the victim's device and sent up to Dropbox cloud automatically. Most likely, these are camera shots that have been named automatically by the device. It would be ideal to sub-group these filenames even further, especially since only some of these images are even related. Thus, the current clustering methodology failed in this part of the scenario.

Node(s) per cluster	Total cluster count
1	303
2	58
3	17
4	13
5	4
6	5
7	2
8	3
10	3
12	2
14	1
15	4
16	2
17	1
18	1
20	1
21	1
24	1
25	2
26	1
28	1
29	1
34	1
39	1
40	1
47	1
54	1
66	1
71	1
77	1
82	1
226	1
699	1
847	1
979	1
1028	1
1052	1
Total no. of clusters	442

Table 4.4: Cluster count based on nodes per cluster for Person B.

Nodes per cluster	File type	Path and Filename pattern
1052	images	/emulated/0/cloudagent/thumbnail/-<10-digit number>
1028	images	cloudagent/cache/Dropbox/Camera Uploads/<date and time>
979	images	/emulated/0/cloudagent/thumbnail/<10-digit number>
847	images	/emulated/0/cloudagent/thumbnail/<9-digit number>
699	images	/emulated/0/cloudagent/thumbnail/-<9-digit number>

Table 4.5: Details of clusters with many nodes for Person B.

File type(s)	Nodes	Cluster count
Word / Excel / Powerpoint	1	6
PDF documents	1	7
PDF documents	2	1
Total number of clusters		14

Table 4.6: Cluster details for PDF, Word, Excel and Powerpoint files.

Now we will see if documents, spreadsheets and powerpoints have been clustered properly or not. We filtered the results based on numerical file type, and the results for the cluster count of PDF, Word, Excel or Powerpoint files are shown in Table 4.6. We see here that most clusters for documents are also single-node clusters. We examined the logs more closely, and it appears that their filenames are completely different from each other, since these files were retrieved from various sources. Thus, the clustering technique has failed here as well.

In addition to the summary, we were also able to make the following observations based on the raw logs:

1. Image files installed by the same app ended up being grouped within the same cluster. This is good, because most of these images are simply image resources of an app. For example, images of avatars for an app.

2. Camera shots that had been uploaded to DropBox cloud from a PC and

downloaded automatically into the Android device are properly clustered based on the event name, since the user applied best-effort in naming his Dropbox folders, so that he can easily select the ones to share with his family or friends. This is also good.

However, the clustering method appears to have failed in the following aspects which are considered more important.

1. All PDF files tend to have a flat structure within the same folder, which would be either Downloads or Documents folder, and the majority appeared as single-node clusters. This result makes sense, because PDF files are normally downloaded by the user into a default folder location, and its filenames were assigned by the corresponding authors.

2. All image shots taken by the camera are placed in the same default folder (e.g. DCIM/), and its filenames are the default auto-assigned names which are based on the date and time that the photo was taken. Because all of these camera shots would have the same filename pattern (date + time) and would be located under the same folder, they would all be grouped within the same cluster based on the calculation rules.

In conclusion, the clustering method has helped in terms of grouping files together to allow us to avoid some redundant file analysis of same-clustered files, but it has not really solved the main problem which is to extract valuable information from the device, since the important files such as PDF files and camera shots have not been clustered properly. The most likely cause of this is that Android users do not normally organize their files on their devices.

4.2 Limited Previewing of PDF files

The 2nd concept is to do limited previewing of PDF files. In reality, this concept can also be extended to other types of documents such as Word, Excel, PowerPoint or even ordinary text files, but for the purposes of this research we shall focus on PDF files. Basically, the main idea is to extract parts of text from a PDF file and determine what the content is about based on the extracted text.

4.2.1 The Scientific Problem of PDF Document Previewing

Let P be a set of pages from the report. Let {X} be a set of the number of characters to be extracted from each page in P. Given P, find {X} where a document can be successfully previewed. The sum of {X} must be kept small if possible.

The following instances have been defined.

1. Let X_1 be the number of characters extracted from the first page P_1.
2. Let X_2 be the number of characters extracted from the middle page P_2.
3. Let X_3 be the number of characters extracted from the last page P_3.
4. Let P_T denote the total number of pages for a document.
5. The page number of the middle page can be calculated by the following formulas: If P_3 is odd, then $P_2 = (P_3 + 1) / 2$. Else $P_2 = P_3 / 2$.

4.2.2 Overview of Methodology

To solve this problem, we will conduct two phases on this experiment. The first phase is to feed in a lot of data into the system and perform test cases using different values of {X}. After a matrix of results has been generated, we select what we think the best values are for {X}, which will then be used in the second testing phase. For the second phase, we used the assigned {X}, and execute the app on unknown devices, and see if we can successfully preview the documents and extract some useful information about the victim. We would confirm the results with the victims afterwards.

4.2.3 Methodology for the 1st Phase

Here are the steps for the first phase:

1. We collected as many PDF files as possible from our existing machines.

2. Used a shell script to rename each PDF file to its corresponding hash value. The purpose is to mask out the names so that our only basis in examining the PDF is the content itself.

3. Feed the PDF into a system where it would extract a set of characters from the top page, the middle page and the bottom page. In this test, we did the following numbers of characters: 50, 100, 150, 200, 250, 400, 600, 800, 1000.

4. For every PDF tested, manually inspect the strings extracted from each page, going from the shortest string to the longest string, stopping at the string where one thinks the data is sufficient enough to provide a good context of the file and decide whether this is valuable enough for stealing.

A total of at least several hundreds of PDFs have been found on our production machine. The entire list has been pre-scanned to remove PDFs that are of same type, for example, if a PDF file F1 is a monthly statement for person X of Bank B, then another PDF file F2 cannot be another monthly statement for the same person X of Bank B even though the dates are different. This is to ensure that the results are generated based on various types of PDF files. Only PDF files written in English language were considered for this experiment.

Filled-in forms require special handling, because we observed that sometimes a PDF writer may place the filled-in text into the PDF structure in one of the following ways:

1. Append new nodes for the filled-in text, thus the best way to retrieve the filled-in text in this case is to extract the last X characters of that page.
2. Filled-in text can be placed in a data structure called form fields. In this case, iText has provided APIs to extract these inputted values. Since this part of our research is more focused on algorithm rather than the technology's features, we will disregard this portion. To analyze this special case, we will assume that if there is enough information extracted from $\{X\}$ that leads to a conclusion of being able to identify the exact form, then this would be sufficient enough to pass the test case.

4.2.4 Experimental Results for the 1st Phase

We filtered and excluded PDF files that cannot be handled properly due to limitations in the library used. We were able to gather a total of 92 PDF files that have been processed properly, and have been selected for further analysis. Out of the 92 files analyzed, 84 files require $X_1 > 0$ to be successfully previewed. X_2 or X_3 may or may not be 0. The results are listed in Table 4.7. The left-column represents the X_1 value required to preview the document. The right column represents the number of documents that require this value of X_1.

We think that if we can cover at least 80% of the cases shown, then the value is good enough. Thus, we have decided to use $X_1 = 400$. In addition to this, further analysis of the raw logs revealed that all documents requiring $X_1 > 400$ to be previewed do not appear to be valuable to the attacker and the victim. We will also use 80% as our benchmark for determining X_2 and X_3 later.

The scattered chart at Figure 4.1 has been plotted to determine if there is any

X_1	Document count
50	5
100	13
150	18
200	17
250	3
400	14
600	6
800	5
1000	3
Total documents	84

Table 4.7: Numbers of documents that require previewing of the first page.

obvious relationship between X_1 and the document's total number of pages. It can be seen that there is no correlation between X_1 and P_T.

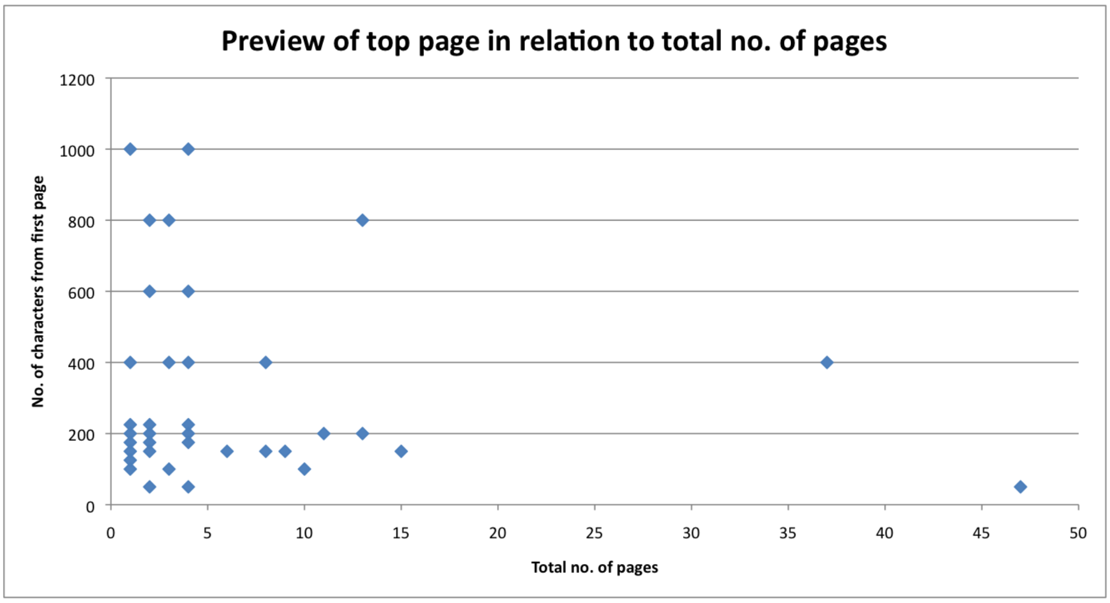


Figure 4.1: Preview of first page in relation to total number of pages.

For the next set of analysis, the results for $X_2 > 0$ will now be shown, for any values of X_1 and X_3 including 0.

Thus, the best value for X_2 is 400 since this value would allow it to cover at least 80% of the documents analyzed.

We have plotted the scattered charts to determine if X_2 depends on either X_1

X_2	Document count
50	1
100	5
150	9
200	3
250	4
400	5
600	1
Total documents	28

Table 4.8: Numbers of documents that require previewing of the middle page.

or P_T. In Figure 4.2, it appears that there is no obvious relationship between X_2 and X_1. In Figure 4.3, X_2 does not depend on P_T. As shown in both charts above, the value of X_2 does not depend on either X_1 or P_T.

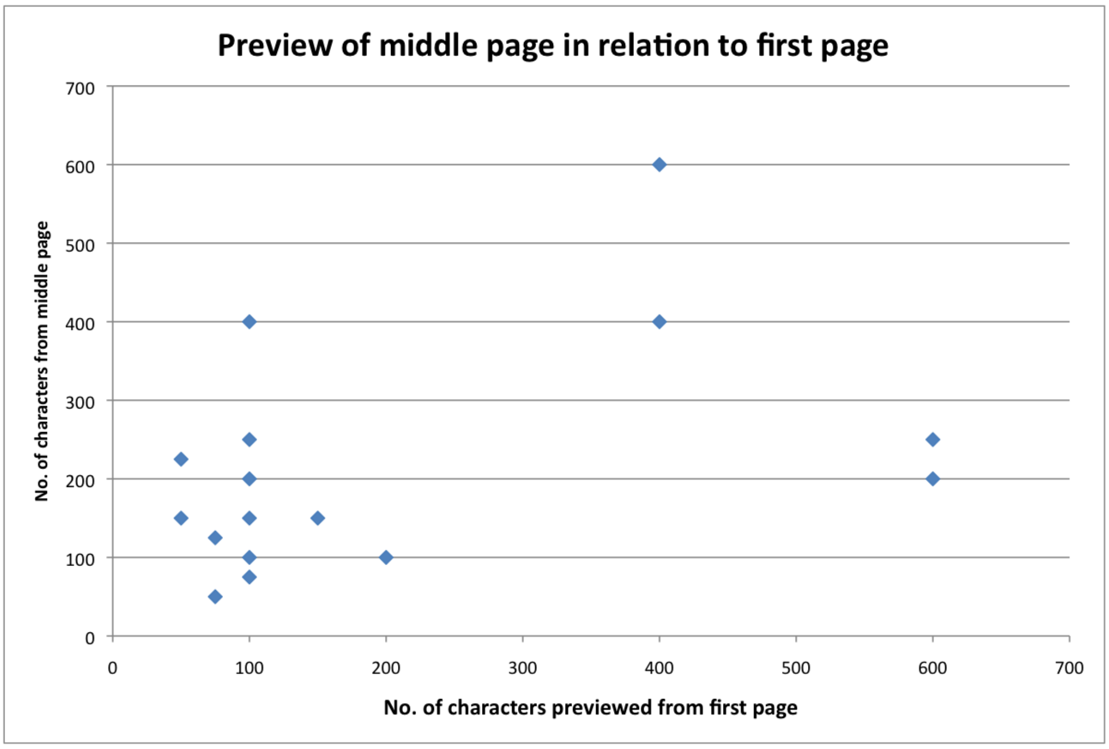


Figure 4.2: Preview of middle page in relation to the first page.

Table 4.9 displays the results for $X_3 > 0$ for any values of X_1 and X_2. By looking at it, it can be easily derived that $X_3 = 250$.

The scattered chart in Figure 4.4 also reveals that X_3 does not depend on the

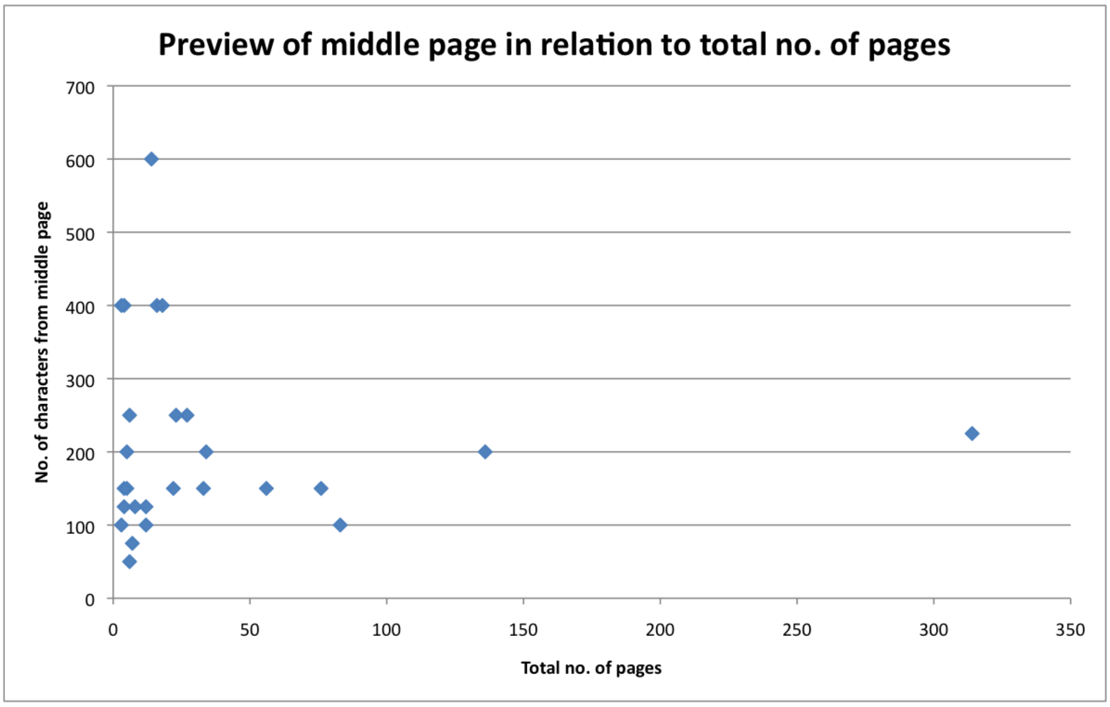


Figure 4.3: Preview of middle page in relation to total number of pages.

X_3	Document count
50	3
100	3
150	0
200	3
250	2
1000	1
Total documents	12

Table 4.9: Numbers of documents that require previewing of at least the last page.

total number of pages.

We also noticed that there are several cases where the text on the top page cannot be a factor in previewing the document. Most likely, some of the documents only use pure images on its 1st page that function simply as the cover page. This tends to be more common among multi-page marketing materials. There are also many cases where previewing the top page on its own is not sufficient enough to get a good context of the document. Therefore, when this is encountered, the

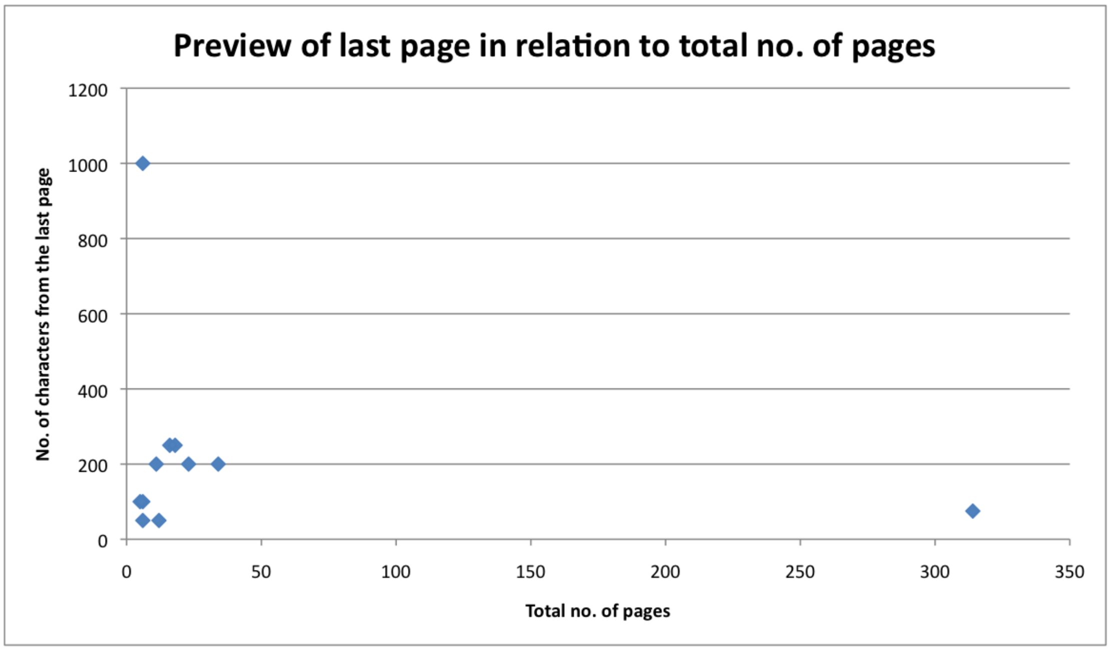


Figure 4.4: Preview of middle page in relation to total number of pages.

middle or bottom page have been previewed as well. But as shown in the charts above, the number of characters previewed on the 1st page does not have any effect on how many characters should be extracted on the other pages if it is required.

4.2.5 Methodology for the 2nd Phase

We proceed with the 2nd phase of this experiment. Assuming that $X_1 = 400$, $X_2 = 400$, and $X_3 = 250$, would it be possible to determine with great confidence if the document is valuable or not based on its predicted context? Also, after all the documents have been previewed, is there enough information to create a profile of the victim?

For this testing phase, we have selected two different users with their primary Android smartphones. Note that they use these phones on a daily basis. They have been asked to execute the app which will perform the following:

1. Traverse the entire SD card to search for PDF files using simplemagic library.

2. If found, do the following:
    (a) Extract the top 400 characters of the 1st page.
    (b) Extract the top 400 characters of the middle page if the document has at least 3 pages.
    (c) Extract the top 250 characters of the last page if the document has at least 2 pages.

4.2.6 Experimental Results for the 2nd Phase

The results of the PDF files found on Person A's device are shown in Table 4.10. A short description about each document found is written in Table 4.11.

ID	No. of pages	Result
[A1]	2	Passed
[A2]	1	Passed
[A3]	6	Passed
[A4]	7	Passed
[A5]	1	Failed
[A6]	5	Passed
[A7]	8	Passed
[A8]	2	Passed

Table 4.10: 2nd phase testing on Person A's device.

Here is the possible story that can be inferred from the results generated. Person A is a Filipino citizen and has recently been admitted to a graduate program at a specific university in Hong Kong, and he intends to enroll. He seems to be trying to learn Cantonese in preparation for his move to Hong Kong.

The story above has been verified with person A, and he has confirmed that all parts of the story are accurate. Thus, the case study for Person A is considered successful.

The results for the PDF files analyzed in person B's device are shown in Table 4.12. A short description for each document is written in Table 4.13.

ID	Description
[A1]	This document appears to be a credit card statement for a specific bank with Person A's full name and the account number listed.
[A2]	This document appears to be a 1-page admission letter from a specific university that has been sent to person A.
[A3]	This document appears to be a multiple page admission letter to the same university mentioned above with detailed payment instructions. The middle page would be of great interest, because theoretically, the attacker can decide to do the following: 1. Get a copy of this document and read it, especially the text used on the middle page. 2. Send a command that can remotely edit the middle page of this document stored on the SD Card, and change the payment instructions to fund the attacker's account instead.
[A4]	A recent product catalog for a specific electronics product. Does not appear to be sensitive material since it appears to be simply a marketing brochure.
[A5]	The previewed characters of the single page document does not clearly define what the document is, although it is almost certain to be education admission related.
[A6]	1st Cantonese lesson guide.
[A7]	2nd Cantonese lesson guide similar to the one above.
[A8]	2-page report on FAQ about travel tax exemption issued by Philippine embassy in Berlin.

Table 4.11: Descriptions of documents scanned on Person A's device.

Based on the results, it is easy to determine that Person B does not have any high-value PDF files that are worth taking. However, it is very easy to derive the following profile for Person B based on the preview results:

Person B appears to be very active in researching about production of livestock feed, such as the many different ways to conduct it and the most recent technological milestones in doing it. Most likely, he has a career in the field of livestock feed production. Person B also has a hobby of cycling.

We find the results of our case study interesting because before this experiment was conducted, we originally thought that person B had a career only in the

ID	No. of pages	Result
[B1]	6	Passed
[B2]	17	Passed
[B3]	9	Passed
[B4]	23	Passed
[B5]	8	Passed
[B6]	1	Passed
[B7]	23	Passed
[B8]	2	Passed
[B9]	25	Passed
[B10]	13	Passed
[B11]	1	Passed
[B12]	4	Passed
[B13]	12	Passed
[B14]	7	Passed

Table 4.12: 2nd phase testing on Person B's device.

ID	Description
[B1]	It is a research paper about producing feed for livestock farming.
[B2]	A research paper about producing renewable energy from parts of sugarcane.
[B3]	A shorter report about the research paper above.
[B4]	Journal report about protein sources for feeding chickens and pigs.
[B5]	A report about farming difficulties.
[B6]	Appears to be a brochure about front-wheel for bicycles.
[B7]	A type of farming technique by using animal waste.
[B8]	A marketing report about a patented air filtration technology.
[B9]	A thick report about plant research.
[B10]	A report about using soybeans as an alternative animal feed.
[B11]	A report about a Thai agency coming up with better production of livestock feed.
[B12]	A report about planting banana trees before raising pigs.
[B13]	A report about using emulsions as an alternative to diesel fuel for machines that are used in livestock feed production engines.

Table 4.13: Descriptions of documents scanned on Person B's device.

wholesale of hardware construction materials and running a grocery store. The results appear to be completely unrelated to the career we thought person B had. But after discussing the results with person B, the following is the correct story about person B:

Person B actually also runs a piggery, and buys ready-made feeds for his pigs. Recently however, he has been considering starting a livestock feed production and see if he can produce and even sell the pig feeds. Person B was quite surprised about the results of this analysis, since he has not really disclosed this plan to other people yet due to financial planning in progress for this potential business idea.

We can observe here that PDF data analysis can go a long way in providing not just the present profile, but also the future intent of the device's user. This can be really helpful for law enforcement for example. If someone appears to be a major threat, then law enforcement agencies can potentially conduct a PDF analysis on the person's Android device, and possibly derive the future intentions or ideas in the works of this person.

4.3 Limited Previewing of PDF Files - An Automated Approach

Another technique that can be considered is automating the process of PDF previewing, and see if the presence of certain keywords can be used to infer the document's category automatically. The objective is to see if there is a set of keywords that are more commonly used within a specific category of documents. This is beneficial in terms of reducing the amount of data transfer required, as only the preview of desired types of documents would be sent up to the server.

There are 3 phases to this technique. The first phase is to select a specific category to predict, and then generate a set of keywords based on known existing data set. The second phase is to determine how many of these keywords are required to consider a document as being of this category. The third phase is to do a live run on other people's devices, and see if document categorization is successful or not based on our formulas and assigned values.

4.3.1 Phase 1: Generating the Keywords

We have decided to select financial as our specific category for the first test case. Based on manual inspection of the previous dataset of 81 files, there are 15 files that are financial-related documents, and 66 files that are not financial-related.

The term *previewed text* refers to the characters that have been extracted from a PDF document based on the following guidelines: top 400 characters of the first page, top 400 characters of the middle page and top 250 characters of the last page. As you may remember, these are the optimal values derived in the previous section.

We created another app that would do the following:

1. Examine each of the previewed text of the 15 financial-related files, and generate a complete list of words that have been used.

2. For each word on the list, let D_F be the number of financial-related documents that have used this word. Let D_N be the number of not financial-related documents that have used this word.

3. Look for all keywords that satisfy both of the following conditions below. These keywords are to be included in the 2nd phase.

   $D_F - D_N > 2$

   $D_F / (D_N + 1) > 2$

We came up with these formulas based on the following desired effects. The first formula ensures that all of the common pronouns or prepositions would be automatically removed, since the number of non-financial documents included in the analysis is much higher than the number of financial documents. The second formula ensures that a keyword must appear many times on the previewed text of financial documents if it were to appear even once or twice on the previewed text of non-financial documents. Because of how the conditions are structured, a qualified keyword must appear on the previewed text of at least 3 financial documents. After performing the analysis above, the list of keywords that meet the criteria are as follows. We define {L} as the set of these keywords.

{L} = { "hkd", "investments", "statement", "thank", "charges", "preferred", "choosing", "summary", "savings", "credit", "fees", "advance", "equivalent", "accounts", "atm", "activities", "balance", "card" }

Table 4.14 displays the detailed analysis of the keywords qualified for the 2nd phase.

Word in {L}	D_F	D_N	D_F-D_N	$D_F/(D_N+1)$
hkd	6	0	6	6
investments	5	0	5	5
statement	9	1	8	4.5
thank	4	0	4	4
charges	4	0	4	4
preferred	4	0	4	4
choosing	4	0	4	4
summary	6	1	5	3
savings	6	1	5	3
credit	6	1	5	3
fees	3	0	3	3
advance	3	0	3	3
equivalent	3	0	3	3
accounts	3	0	3	3
atm	3	0	3	3
activities	3	0	3	3
balance	8	2	6	2.67
card	5	1	4	2.5

Table 4.14: List of keywords qualified for categorizing a document to be financial.

4.3.2 Phase 2: Determining the Unique Keyword Count

Let W be the number of unique keywords required from {L} that must be found in the previewed text of a document for it to be considered financial.

Given {L}, we need to derive W. Another app has been created to look at the previewed text of each PDF file, and it will count the number of unique keywords from {L} that are found in the previewed text. We downloaded another 165 random PDF documents from the Internet that are not financial. These files are then copied to the SD Card of the Android device, and then the app is executed. Table 4.15 displays a summary of the result. Based on the summary, since only one file had 2 unique keywords on its previewed text, we can set W = 3.

No. of unique keywords found	No. of non-financial documents
0	144
1	20
2	1

Table 4.15: Count of non-financial documents based on how many unique keywords were found in the previewed text of each document.

4.3.3 Phase 3: Testing the Keywords

Another app has been created to analyze the previewed text of all PDF files on the SD Card of an Android device, and determine which ones are considered financial given that W = 3. The app has been executed on the devices of the same Person A and Person B from the previous section. The results are shown in Table 4.16 and Table 4.17.

ID	No. of pages	Score	Predicted category	Result
A1	2	4	Financial	Correct
A2	1	0	Not financial	Correct
A3	6	1	Not financial	Correct
A4	7	0	Not financial	Correct
A5	1	1	Not financial	Correct
A6	5	0	Not financial	Correct
A7	8	0	Not financial	Correct
A8	2	0	Not financial	Correct

Table 4.16: Categorization of documents in Person A's device based on unique keywords found in each of the previewed text.

ID	No. of pages	Score	Predicted category	Result
B1	6	0	Not financial	Correct
B2	17	0	Not financial	Correct
B3	9	0	Not financial	Correct
B4	23	0	Not financial	Correct
B5	8	1	Not financial	Correct
B6	1	0	Not financial	Correct
B7	23	0	Not financial	Correct
B8	2	0	Not financial	Correct
B9	25	0	Not financial	Correct
B10	13	0	Not financial	Correct
B11	1	0	Not financial	Correct
B12	4	0	Not financial	Correct
B13	12	0	Not financial	Correct
B14	7	0	Not financial	Correct

Table 4.17: Categorization of documents in Person B's device based on unique keywords found in each of the previewed text.

It can been seen from the results that this technique has been successful in identifying which documents are financial. To summarize, we have determined that 1 out of 22 PDF files previewed on both devices is a financial document. All of the results have been confirmed to be correct.

4.3.4 Another Test Case

We have proven that our proposed algorithm works for categorizing financial documents. We will now try another test case, where our specific category will be education. More specifically, the topics that will be considered related to education are information about different universities and their corresponding programs, admissions, grades and procedures.

1. **Generating the Keywords.** Out of the 81 files in the previous dataset, we have determined based on manual inspection that 9 files are documents about education. Following the same technique in the previous test case, the list of keywords and their corresponding computations are displayed in

Table 4.18. Based on the analysis, we define the following:

{L} = { "admission", "college", "mba", "graduate", "institutions", "institution", "postgraduate" }

2. **Determining the Unique Keyword Count.** Following the same set of procedures in the previous test case, we apply {L} to our app and execute it against a set of 163 PDF files randomly downloaded that are not related to education. Table 4.19 provides a summary of the unique keyword count. Based on the results, we can set W = 2.

3. **Testing the Keywords.** Given W = 2, the final test has been performed on Person A's and Person B's devices to determine which of their PDF files are about education. As shown in Table 4.20 and Table 4.21, we missed one document which has been incorrectly categorized as not related to education. However, the test is still mostly successful. Most likely, we need to increase the data set in the first phase so that more keywords can be generated to have better coverage of documents related to education.

Word in {L}	D_F	D_N	D_F-D_N	$D_F/(D_N+1)$
admission	4	0	4	4
college	3	0	3	3
mba	3	0	3	3
graduate	3	0	3	3
institutions	3	0	3	3
institution	3	0	3	3
postgraduate	3	0	3	3

Table 4.18: List of keywords qualified for categorizing documents as related to education.

No. of unique keywords found	No. of education documents
0	155
1	8
2	0

Table 4.19: Count of non-education documents based on how many unique keywords were found in the previewed text of each document.

ID	No. of pages	Score	Predicted category	Result
A1	2	0	Not education	Correct
A2	1	1	Not education	Incorrect
A3	6	2	Education	Correct
A4	7	0	Not education	Correct
A5	1	2	Education	Correct
A6	5	0	Not education	Correct
A7	8	0	Not education	Correct
A8	2	0	Not education	Correct

Table 4.20: Categorization of documents in Person A's device based on unique keywords found in each of the previewed text.

ID	No. of pages	Score	Predicted category	Result
B1	6	0	Not education	Correct
B2	17	1	Not education	Correct
B3	9	0	Not education	Correct
B4	23	0	Not education	Correct
B5	8	1	Not education	Correct
B6	1	0	Not education	Correct
B7	23	0	Not education	Correct
B8	2	0	Not education	Correct
B9	25	0	Not education	Correct
B10	13	0	Not education	Correct
B11	1	0	Not education	Correct
B12	4	0	Not education	Correct
B13	12	0	Not education	Correct
B14	7	0	Not education	Correct

Table 4.21: Categorization of documents in Person B's device based on unique keywords found in each of the previewed text.

CHAPTER 5

Conclusion and Future Work

5.1 Conclusion

In conclusion, targeted attacks can also be successfully conducted on Android. Due to the availability of features on an Android smartphone that are not available on Windows, targeted attacks on Android can be considered even more dangerous, as it allows access to a wide variety of potentially sensitive information, such as SMS messages, call logs or contact records. The constrained environment such as limited network bandwidth would not be effective in preventing information gathering, as we have demonstrated that limited previewing of PDF files make it possible for someone to gather a good amount of information and even perform profiling on the victim. We have also demonstrated the potential of using automation for PDF previewing to simplify the analysis process. However, clustering based on path names and filenames would not be a good technique in its current form.

5.2 Future Works

There is so much more room for this research to expand on. On the implementation of the targeted attack, here is a list of other features that we believe can be added. On the Android app that was retargeted from the Windows version of Greencat, some of these functionalities can be included to demonstrate the extent of potential damages.

1. Editing contacts on address book
2. Editing existing SMS messages
3. Inserting and deleting phone call logs.
4. Intercepting new SMS messages
5. Sending new SMS messages without user knowledge
6. Setting up an auto-start mechanism

Other features that can be implemented to make the app more usable are as follows:

1. Compression of files using different available compression schemes on Java before they are uploaded to the server to reduce propagation time
2. Using a different protocol other than HTTP since base64 encoding and decoding is not very efficient due to expansion of size before the transfer is conducted. But this change would also be required on the server component.

In regards to clustering, one can also improve the way it would handle images. For example, as we have noticed, the date and time is used in automatic naming of the image files where the camera shots are stored. We can potentially use this information as part of the clustering algorithm by grouping images of similar date/time together.

We have shown that it is possible to add automation so that PDF previewing can make a determination on the document's category, which helps automatically assess if a document can possibly be valuable or not. However, the formulas and variables used can be improved further by adding more data to the sample set. This would allow better fine tuning of the values and conditions.

References

[Hen09] S. Henderson. "How Do People Manage Their Documents? An empirical investigation into personal document management practices among knowledge workers." (Dissertation). University of Auckland, 2009.

[Wue14] C. Wueest. "Targeted Attacks Against the Energy Sector." https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/targeted_attacks_against_the_energy_sector.pdf, Jan 13 2014.

[Pro11] S. Proske. "Computer Invaders: The 25 Most Infamous PC Viruses of All Time." http://safeandsavvy.f-secure.com/2011/03/21/25-infamous-viruse/#.U29uOq2Sw71, Mar 21 2011.

[Idc13] IDC Press Release. "Tablet Shipments Forecast to Top Total PC Shipments in the Fourth Quarter of 2013 and Annually by 2015, According to IDC." http://www.idc.com/getdoc.jsp?containerId=prUS24314413, Sept 11 2013.

[FG13] C. Funk and M. Garnaeva. "Kaspersky Security Bulletin 2013. Overall statistics for 2013." https://www.securelist.com/en/analysis/204792318/Kaspersky_Security_Bulletin_2013_Overall_statistics_for_2013, Dec 10 2013.

[RM14] J. Rivera and R. van der Meulen. "Gartner Says Annual Smartphone Sales Surpassed Sales of Feature Phones for the First Time in 2013." http://www.gartner.com/newsroom/id/2665715, Feb 13 2014.

[Tre12] Trend Micro Incorporated. "Research Paper 2012: Adding Android and Mac OS X Malware to the APT Toolbox." http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/white-papers/wp_adding-android-and-mac-osx-malware-to-the-apt-toolbox.pdf, taken May 10 2014.

[Jet11] L. Jeter. "Review of the Usage of Security Mechanisms Within the Android Operating System." (Master's thesis). University of Colorado, 2011.

[JCI11] The JCIFS Project. "The Java CIFS Client Library." http://jcifs.samba.org/, Oct 18 2011.

[Wat14] G. Watson. "SimpleMagic - Simple Magic Content and Mime Type Detection Java Package." http://256.com/sources/simplemagic/, Apr 28 2014.

[MSD14] Microsoft Corp. "Operating System Version (Windows)." http://msdn.microsoft.com/en-us/library/windows/desktop/ms724832(v=vs.85).aspx, taken May 1 2014.

[Ite14] iText Software Corp. "iText 5.4.5 | iText Software." http://itextpdf.com/release/iText545, Dec 2 2013.

[Sym14] Symantec Corp. "Advanced Persistent Threats: A Symantec Perspective." http://www.symantec.com/content/en/us/enterprise/white_papers/b-advanced_persistent_threats_WP_21215957.en-us.pdf, taken May 13 2014.

[Go14a] Google Inc. "Android Developers - FileObserver." http://developer.android.com/reference/android/os/FileObserver.html, taken May 24 2014.

[Go14b] Google Inc. "Android Developers - ContentObserver." http://developer.android.com/reference/android/database/ContentObserver.html, taken May 24 2014.

[Go14c] Google Inc. "Android Developers - Browser." http://developer.android.com/reference/android/provider/Browser.html, taken May 24 2014.

[Go14d] Google Inc. "Android Developers - Manifest.permission." http://developer.android.com/reference/android/Manifest.permission.html, taken May 24 2014.