A First Look at Android Malware Traffic in First Few Minutes

Zhenxiang Chen*, Hongbo Han*, Qiben Yan†, Bo Yang*, Lizhi Peng*, Lei Zhang*, and Jin Li‡
*Shandong Provincial Key Lab of Network based Intelligent Computing
University of Jinan
Jinan, Shandong, China, 250022
†Shape Security
Mountain View, CA, USA, 94040
‡Guangzhou University
Guangzhou, Guangdong, China, 510006

***Abstract*—With the advent of mobile era, mobile terminals are going through a trend of surpassing PC to become the most popular computing device. Meanwhile, the hackers and virus-writers are paying close attention to the mobile terminals, especially the Android platform. The growing of malwares on the Android system has drawn attentions from both the academia and security industry.Recently, mobile network traffic analysis has been used to identify the malware. But due to the lack of a large-scale malware repository and a systematic analysis of network traffic features, the existing research mostly remain in theory. In this paper, we design an Android malware traffic behavior monitoring scheme to capture traffic data generated by malware samples in a real Internet environment. We capture the network traffic from 5560 malware samples in the first 5 minutes, and analyze the major compositions of the traffic data. We discover that HTTP and DNS traffic are accounted for more than 99% on the application layer traffic. We then present an analysis of related network features: DNS query, HTTP packet length, ratio of downlink to uplink traffic amount, HTTP request and Ad traffic feature.Our statistical results illustrate that: (1) more than 70% malwares generate malicious traffic in the first 5 minutes; (2) DNS query and HTTP request can be used to identify the malware, and the detection rate reaches 69.55% and 40.89% respectively; (3) Ad traffic can greatly affect the malware detection. We believe our research provides an in-depth analysis into mobile malwares' network behaviors.**

***Keywords*-malware; traffic behavior; traffic analysis;**

I. Introduction

The Internet usage of mobile phones has already surpassed that of PC. Meanwhile, the security problem of mobile phone has become a general concern of the industry. Given the rapid growth of Android malwares, there is a pressing need to mitigate or defend against them effectively. Researchers have spent a significant amount of efforts to characterize malwares on mobile application markets [1]. Most of them have applied static or dynamic analysis techniques on a large number of applications in an attempt to identify malwares.

However, to provide real time analysis of application behaviors are too resource consuming to be deployed on smartphones. Also, despite all of the existing efforts, the extent to which the mobile ecosystem been corrupted by mobile malwares is not well understood. Recently, some researchers begin to focus on network level analysis of mobile malwares using traffic from a wireless access network or cellular network. Nevertheless, without a deep understanding of the malware traffic characteristics, it is hard to develop an effective and practical mitigation solution. To make things worse, the research community at large is still constrained by the lack of a comprehensive mobile malware traffic dataset to start with.

In this paper, we report our results on the collection and characterization of Android malware traffic on smartphone. We employ active traffic generator and passive sniffers on the network to record all inbound and outbound traffic. Given the legal consequences of actively deploying a large number of malware samples on the real network environment, we must control the risk strictly. The results in this paper are based on the datasets from a recent study called Drebin [2], which consists of 5560 malware samples (177 families). We deploy and capture the packet-level traces in the first few minutes to get a traffic dataset of about 500.4MB size. Then, we dig more deeply into the traffic dataset and uncover a number of important insights and features regarding malicious traffic behaviors. This paper mainly makes the following contributions:

- Design and implement an Android malware traffic generation and collection scheme.
- Use a real world dataset of 5560 malware samples to capture their network traffic traces.
- Make a first look at the Android malware traffic in the first few minutes and analyze its characteristics.
- Get a list of Android malwares target IP, CNAME, malicious domains, generate the blacklist and prepare for malware detection.

II. Methodology

To capture and analyze Android malware network traffic, we design a malware traffic generation and capturing platform. Our method employs a traffic monitor platform to monitor traffic, and then run a large number of Android malware samples on our platform. We find some key elements that will affect the traffic generation, based on which we design automated traffic generation and collection algorithm. In the following sections, we will discuss the methodology in detail.

Table 1: Top 24 families in our dataset(number $\geqslant$ 25)

ID	FamilyName	Num	ID	FamilyName	Num
1	FakeInstaller	925	13	ExploitLinuxLotoor	70
2	DroidKungFu	667	14	Glodream	69
3	Plankton	625	15	MobileTx	69
4	Opfake	613	16	FakeRun	61
5	GinMaster	339	17	SendPay	59
6	BaseBridge	330	18	Gappusin	58
7	Iconosys	152	19	Imlog	44
8	Kmin	147	20	SMSreg	41
9	FakeDoc	132	21	Yzhc	37
10	Geinimi	92	22	Jifake	29
11	Adrd	91	23	Hamob	28
12	DroidDream	81	24	Boxer	27

A. Andriod Malware Dataset

In our experiment, we use a dataset of real world Android malwares. The dataset comes from Drebin project [2]. Additionally, it includes all samples from the Android Malware Genome Project [1]. After removing the Adware samples, the final dataset contains 5,560 malware samples. An overview of the top 24 malware families(malware number excess 20) in the dataset is provided in Table I including several families that are still being actively distributed in application markets. Note that only the top 24 families are displayed.

B. Traffic Monitor Platform

In order to get malware traffic traces in a real network environment, we design an active traffic generation and monitoring platform. As shown in Figure 1, the platform consists of four parts: foundation platform, traffic generator, traffic collector and network proxy/firewall.

Foundation platform is based on Android Virtual Device (AVD) [3], with the debugging tool as Android Debug Bridge(ADB). In the meantime, we provide *Samsung Glaxy S*2 to capture traffic and deal with the abnormal operation Apps. This foundation platform provides a basic Android simulation environment and command line mode of interaction, and it could realize some basic functionalities: creation, installation and operation.

The traffic generator is designed to install and activate malware samples to generate traffic traces automatically, which consists of two components: automatic traffic generator and malware execution controller. Automatic traffic generator implements automatic installation and activation of the malware. Malware execution controller uses monkeyrunner [4] that has an API to control the malware execution path.

The traffic collector is designed to capture the inbound and outbound traffic automatically using tcpdump, and we employ traffic mirror technology to mirror traffic that pass through the gateway to a server.

A network transport proxy/firewall was deployed between the Internet and malware running environment to monitor and control the attack behavior.

C. Traffic Generation Analysis

By analysis, we find that installation, INTERNET permission, activation and aborting are four key elements that will directly affect the malware running and traffic generation

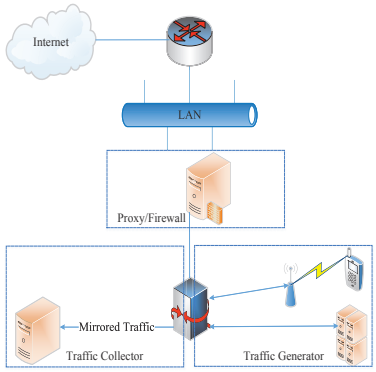


Figure 1: Malware traffic generation and capturing platform

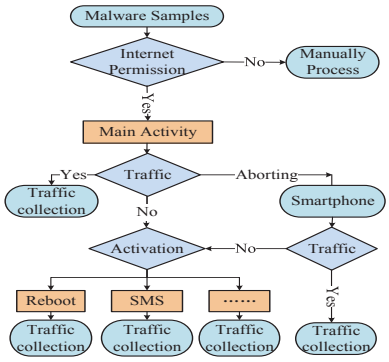


Figure 2: The automatic malware traffic generation workflow

behaviors. The top level work flow of traffic generation is shown in Figure 2. More details are illustrated as follows.

1) Installation: The existing methods Android malwares used to be installed into users' phones can be categorized into three main social engineering techniques, i.e., repackaging, update attack and drive-by download [1]. For most of malwares, the malware installation happens before the malware traffic generation, except for a small portion of the malware which only includes an update component that will fetch or download the malicious payloads in runtime.

2) The INTERNET Permission: For Android Apps without root exploits, their capabilities are strictly constrained by the permissions users grant to them.The statistical result of permissions for the selected 5560 samples is shown in Figure 3. Here we list the top 15 of most commonly used sensitive permissions. As we expected, 5344 samples (96.2% of 5560 samples) use the INTERNET permission, only 216 samples (3.8% of 5560 samples) are not requesting the INTERNET permission in our dataset.

3) Activation: Malwares can activate themselves in several ways, among all available system events, BOOT_COMPLETED is the most popular one to existing Android malware, and 83.3% of the total 1026 samples listen to this event [1]. This is not surprising as this particular event will be triggered when the system finishes its booting process which is a perfect time for malware to kick off its background services. This is one of the main

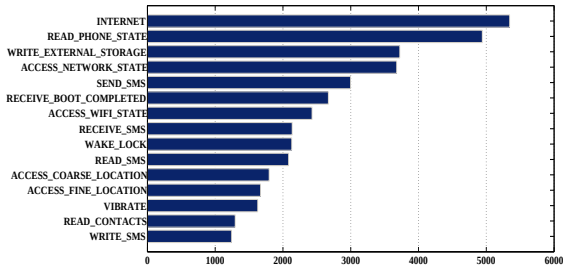


Figure 3: The top 15 permissions requested by 5560 malware samples

reasons that we look into the malware traffic only in the first minutes. Also, the activation method used by most of malwares in our dataset is BOOT_COMPLETED.

4) Aborting: Running a malware on the AVD, aborting is the main abnormal phenomenon which affects Internet traffic generation. There are about 200 samples (3.6% of all 5560 samples) aborting their execution immediately after installation. These malware are installed on $Samsung\ Glaxy\ S2$, which is based on Android 2.3. However, most of them still abort when running. After checking their APK files, we find that the main reason is that some malwares cannot run on Android 2.3, such as $DroidDeluxe$.

Based on the analysis above, for the elements of non-active, aborting and some other elements such as failed installation and failed execution etc., we have no specific collection mechanisms. We will improve the traffic collection platform to consider those elements in the future work.

D. Automatic Traffic Generation and Collection

In order to capture 5560 malware samples' traffic automatically, we need an automated script to implement the automatic installation, operation, activation and collection. Based on the analysis above, we design an automated traffic generation and collection algorithm. Primarily, we need to do some preprocessing work, includes decompiling APK files to get the package and activity name, then writing APK name, package name and activity name into a txt file line by line. More details are shown in Algorithm 1.

Data: the malware in the same family
Result: capture the traffic generated by the malware
while *not at the end of the txt file* **do**
  read the txt file line by line to get malware name, package name and activity name;
  if *the malware APK file, package name and activity name all exist* **then**
    create an AVD;
    start up AVD and install the malware;
    reboot this emulator;
    read the package name and activity name to start up the malware;
    start tcpdump to capture traffic;
    delete this AVD and move to the next;
  else
    write the error information into the log file;
  end
end

Algorithm 1: The automatic traffic generator algorithm

III. Experiment and Result

A. Basic Statistics of Traffic Data Set

Based on the designed platform, we collected the network traffic of 5560 malware samples in first 5 minutes. The captured raw data include many irrelevant traffic, such as SSDP, DHCP, ARP, NBNS, IGMP, SMB, which are all removed by applying the Wireshark filters. The basic traffic statistical information of the top 24 families as in Table I, is shown in Table II. Each of these top 24 families has sample number $\geqslant 25$, with a total of 4785 samples (86.06% of all). Further, we analyze the basic traffic breakdown by the application layer protocols which is shown in Table III, and find that SSL only take a very small part. The major traffic types are HTTP (89.62%) and DNS (10.24%) traffic.

Table III: DNS/HTTP/SSL traffic proportion in the top 24 families

FamilyName	DNS	HTTP	SSL
FakeInstaller	0.78%	15.59%	0.19%
DroidKungFu	0.25%	3.54%	0.00%
Plankton	2.03%	26.07%	0.00%
Opfake	11.01%	25.27%	0.63%
GinMaster	5.45%	16.38%	0.01%
BaseBridge	3.66%	24.15%	0.00%
Iconosys	5.40%	24.31%	0.00%
Kmin	4.52%	5.50%	0.00%
FakeDoc	0.90%	7.11%	0.00%
Geinimi	7.58%	19.22%	0.00%
Adrd	4.36%	31.37%	0.23%
DroidDream	3.83%	10.54%	0.00%
ExploitLinuxLotoor	0.60%	3.56%	0.00%
Glodream	4.60%	15.76%	0.00%
MobileTx	1.52%	34.53%	0.00%
FakeRun	4.19%	32.07%	0.00%
SendPay	4.22%	29.20%	0.00%
Gappusin	1.75%	11.73%	0.00%
Imlog	3.68%	18.30%	0.00%
SMSreg	0.41%	3.78%	0.00%
Yzhc	10.09%	0.07%	0.00%
Jifake	0.47%	0.08%	0.00%
Hamob	0.92%	9.37%	0.00%
Boxer	1.47%	5.80%	0.00%

Zhou et al. [1] examines the remote control functionality of the malware payloads. They find that 93.0% of the samples turn the infected phones into bots for remote control. Specifically, almost all of these samples that use the HTTP-based web traffic to receive bot commands from their C&C servers. Furthermore, they also find that most C&C servers are registered in domains controlled by attackers themselves. So we will primarily focused on the DNS and HTTP traffic in this paper.

B. DNS Traffic Analysis

In this section, we statistically analyze the top 24 families' DNS queries. For simplicity, we select the top

Table II: Basic traffic data information in the top 24 families; The "RS" represents the raw traffic data size, "PN" represents the packet number after preprocessing, "PS" represents the corresponding packet size, "AL" represent the average length, "OIN" represents the total number of outbound and inbound packets, "TS" represents the corresponding total size, "ON"("IN") represents the outbound (inbound) packets number, "OS" ("IS")represents the corresponding outbound (inbound) packets size, "OAL" ("IAL") represents the outbound (inbound) packets average length.

FamilyName	RS(byte)	Preprocess Data			Outbound and Inbound Data		Outbound Data			Inbound Data		
		PN	PS(byte)	AL(byte)	OIN	TS(byte)	ON	OS(byte)	OAL(byte)	IN	IS(byte)	IAL(byte)
FakeInstaller	28891098	5791	2661096	919.05	5606	2638148	2670	328033	245.72	2936	2310115	1573.65
DroidKungFu	241206621	244015	220744514	1809.27	240296	220211261	81007	9071299	223.96	159289	211139962	2651.03
Plankton	47837937	111577	29613926	530.82	107514	29009385	57761	10074698	348.84	49753	18934687	761.15
Opfake	18145471	7085	1724674	486.85	4512	1534742	2296	383653	334.19	2216	1151089	1038.89
GinMaster	22251568	23701	7330829	618.61	19977	6935946	10460	1866191	356.82	9517	5069755	1065.41
BaseBridge	15924586	12552	3527586	562.08	11549	3398188	6739	1279561	379.75	4810	2118627	880.93
Iconosys	5410930	1296	264282	407.84	1158	249947	689	107116	310.93	469	142831	609.09
Kmin	7281927	8507	2787177	655.27	7580	2658656	3957	574358	290.3	3623	2084298	1150.59
FakeDoc	27087667	45068	22667981	1005.95	43403	22465052	19953	2477636	248.35	23450	19987416	1704.68
Geinimi	3602595	2168	405290	373.88	1970	374578	1137	164238	288.9	833	210340	505.02
Adrd	4070859	4747	916891	386.3	4265	839238	2551	415861	326.04	1714	423377	494.02
DroidDream	3122966	1392	536753	771.2	1236	516410	609	87265	286.58	627	429145	1368.88
ExploitLinuxIotoor	3249874	2191	1381472	1261.04	2143	1373204	930	115500	248.39	1213	1257704	2073.71
Glodream	2813616	2560	800788	625.62	2342	763967	1257	182634	290.59	1085	581333	1071.58
MobileTx	4650799	7310	2504225	685.15	6993	2466264	4198	779809	371.51	2795	1686455	1206.77
FakeRun	3321622	4774	928055	388.8	4488	889151	2503	453336	362.23	1985	435815	439.11
SendPay	2542003	1531	274828	359.02	1415	263100	864	127448	295.02	551	125652	492.38
Gappusin	5055953	6969	2955707	848.24	6610	2904108	3343	494288	295.72	3267	2409820	1475.25
Imlog	2214692	1098	325723	593.3	992	313752	528	92449	350.19	464	221303	953.89
SMSreg	4959810	5964	3765050	1262.59	5876	3749018	2381	291571	244.91	3495	3457447	1978.51
Yzhc	1789093	1617	687623	850.49	764	618031	277	25478	183.96	487	592553	2433.48
Jifake	1183356	507	331762	1308.73	495	330074	199	23904	240.24	296	306170	2068.72
Hamob	9417301	14470	8599747	1188.63	14061	8520617	5811	823083	283.28	8250	7697534	1866.07
Boxer	758419	284	170491	1200.64	270	167980	118	13919	235.92	152	154061	2027.12
$Total$	466790763	517174	315906470		495515	313190817	212238	30253328		283277	282937489	

3 queries in the same family, which account for more than 80% of all. We use the URLvoid [5] includes Dr.Web, TrendMicro, AVGThreatLabs and so on more than 40 kinds, which is more accurate than a single scanning engine. Detection results is shown in Table IV, "$\surd$" means the query is detected as a malicious query. Here we give several examples($FakeInstaller$,$DroidKungFu$,$Plankton$, $Opfake$,$GinMaster$) in 5 families (3169 samples, 56.996% of all 5560). For example, $waply.ru$ in $FakeInstaller$, $app.wapx.cn$ in $DroidKungFu$, $www.apperhand.com$, $api.airpush.com$, $ad.leadboltapps.net$ in $Plankton$, $gaga01.net$, $m-001.net$ in $Opfake$, $client.go360days.com$ in $GinMaster$, they account for more than 85.67% of all queries that have been detected as malicious queries. Actually, $gaga01.net$ has already been confirmed to spy on the phone IMEI and other personal information through connecting back to the $gaga01.net/rq.php-93.170.107.57-Email$. The other one is the $client.go360days.com$ in $GinMaster$. As TrustGo announced in 2012, it was one of the remote control servers. But on the other hand, some DNS queries that have not been detected may also be generated by malwares. For example, $mobile.tx.com.cn$ and $tx.com.cn$ in $MobileTx$, in September 2011, are detected as an new Android riskware by F-Secure, which will send user's IMSI to the given servers. As a statistical result, the DNS queries can be used to help identify the malware behavior as a minimum statistical probability of 69.55%.

C. HTTP Feature Analysis

1) HTTP Packet Length Analysis: According to the packet lengths, HTTP packets can be divided into several intervals as $20-39, 40-79, 80-159, 160-319, 320-639, 640-1279, 1280-2559, 2560-5119$ and $5120-$. We analyze the HTTP packet length features (the distribution at various intervals, average length, maximum length and minimum length) in the top 24 families. Figure 4 shows the distribution of the HTTP packet length within different intervals. The interval follows a normal distribution, and most of the HTTP traffic has the interval of $320-639$ (at about 45%).

Next, we calculate the expectation and variance in the same family. With respect to different intervals, we analyze the packet number (which can be set as n_i), and the total packet number (set as N_i), where i means the ith family. The probability of the sample is $p_i = \frac{n_i}{N_i}$, the sample mean in the same interval is the sample value, which can be denoted as $X_{ij} = \frac{\sum_{k=1}^{n_i} x_{nk}}{n_i}$, j means the jth interval, $\hbar$ means the total number of the intervals. The expectation $E(X_i)$ can be calculated according to the following formula:

$$E(X_i) = \sum_{j=1}^{\hbar} p_i \cdot X_{ij}$$

After getting the expectation $E(X_i)$, the variance $D(X_i)$ can be calculated as:

$$D(X_i) = \frac{1}{\hbar - 1} \cdot \sum_{j=1}^{\hbar} (X_{ij} - E(X_i))^2$$

Square root $D(X_i)$, we get the standard deviation $\sigma_i = \sqrt{D(X_i)}$, in Table V. We calculate the total 24 families' HTTP length expectation and variance as 561.41 and 351.98, which could be used as a benchmark to identify the malware.

2) Ratio of Downlink to Uplink Traffic Amount Analysis: Downlink traffic means the response from the server to the emulator(inbound data), and relatively uplink traffic means the malware requests to the server(outbound data). As a network behavior feature, we compute the ratio of downlink to uplink traffic in Figure 5. PRatio represents the data packet number ratio of downlink to uplink, BRatio represents the byte ratio of all received to all sent data, x axis represent the

Table IV: An overview of the top 3 DNS queries and identified malicious queries in the top 24 families

FamilyName	Top query 1	Result	Top query 2	Result	Top query 3	Result
FakeInstaller	waply.ru	√	browser-error-page.ru	√	i.yangruiling.com	
DroidKungFu	app.wapx.cn	√	req.adsmogo		appsrv1.madserving.cn	
Plankton	www.apperhand.com	√	api.airpush.com	√	ad.leadboltapps.net	√
Opfake	gaga01.net	√	001.net			
GinMaster	client.go360days.com	√	cn.papayamobile.com		connect.papayamobile.com	
BaseBridge	dev.adtouchnetwork.net	√	b3.8866.org	√	b4.cookier.org	
Iconosys	smsreplier.net	√	blackflyday.com		graph.facebook.com	
Kmin	transit.5kzk.com		transit.5j5w.com		jujoy.5y3g.com	
FakeDoc	chart.googleapis.com		moba.rsigma.com		bongacams.com	√
Geinimi	google.funimoe.com		www.winpowersoft.com		data.flurry.com	
Adrd	log.android188.com		log.meego91.com	√	adrd.taxuan.net	√
DroidDream	www.gstatic.com		kiu6.com		mm.admob.com	
ExploitLinuxLotoor	www.umeng.com		r.admob.com	√	mm.admob.com	
Glodream	www.gstatic.com		cfg.adsmogo.mobi		cfg.adsmogo.com	
MobileTx	mobile.tx.com.cn		tx.com.cn		img.tx.com.cn	√
FakeRun	ad.leadboltapps.net	√	api.airpush.com	√	www.droidsettings.com	
SendPay	api.go108.cn		d.wiyun.com			
Gappusin	app.wapx.cn	√	ads.wapx.cn	√	www.umeng.com	
Imlog	wp.ysler.com		www.imnet.us	√	waps.ysler.com	
SMSreg	api.airpush.com	√	ro.plus1.wapstart.ru	√	www.apperhand.com	√
Yzhc	domaindev.51widgets.com		admin.51widgets.com	√	axy.waplove.cn	√
Jifake	crl.globalsign.com					
Hamob	stat.appsgeyser.com		ads.appsgeyser.com		rq.vserv.mobi	
Boxer	media.admob.com		www.gstatic.com		data.flurry.com	

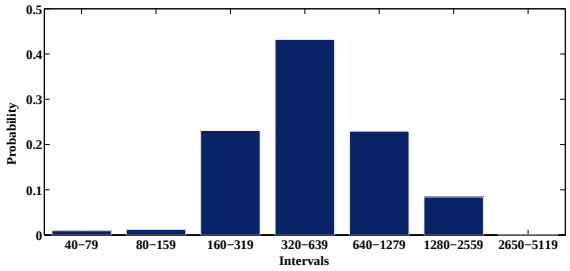


Figure 4: The HTTP packet length distribution of the top 24 families

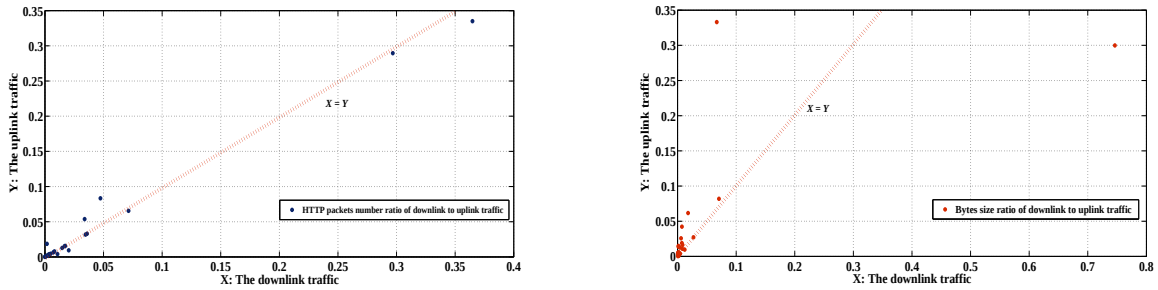
Table V: The average length/expectation/variance/standard deviation of HTTP packets in the top 24 families

FamilyName	Average Length(byte)	$E(X_i)$	$D(X_i)$	σ_i
FakeInstaller	1346.64	673.32	307650	554.66
DroidKungFu	1119.68	559.84	108900	330
Plankton	1253.40	626.70	121530	348.61
Opfake	2013.24	1006.62	218920	467.89
GinMaster	1143.84	571.92	300620	548.29
BaseBridge	914.20	457.10	26961	164.20
Iconosys	1019.98	509.99	134240	366.39
Kmin	692.09	346.04	94061	306.69
FakeDoc	1121.69	560.85	73923	271.89
Geinimi	973.61	486.81	105170	324.30
Adrd	828.81	414.40	34102	184.67
DroidDream	1854.79	927.40	343040	585.70
ExploitLinuxLotoor	848.28	424.14	100110	316.40
Glodream	882.67	441.33	98242	313.44
MobileTx	1208.51	604.26	56254	237.18
FakeRun	1031.57	515.79	53806	231.96
SendPay	810.60	405.30	10993	104.85
Gappusin	1028.77	514.39	64404	253.78
Imlog	1241.73	620.87	78276	279.78
SMSreg	837.80	418.90	127550	357.14
Yzhc	506	253	0	0
Jifake	506	253	0	0
Hamob	1230.60	615.30	119380	345.51
Boxer	1412.29	706.14	141140	375.69
$Total$	1076.12	574.79	123260	351.08

downlink traffic, y axis represent the uplink traffic, the red line represents $x = y$, means the downlink traffic equals to the uplink traffic. Comparing Figure 5(a) with Figure 5(b), we find the byte size ratio in most families is above the red line in Figure 5(b), which means uplink traffic amount is greater than downlink traffic amount,which also means most malware upload data more frequently with larger traffic volume. However, in Figure 5(a), we can see that the packet number ratio in most families is growing along with the red line.

3) HTTP Request Analysis: To analyze the compositions of the HTTP traffic, similar to DNS traffic analysis, we select the top 5 HTTP requests of the top 24 families. First, we still use URLvoid [5] as the malicious detection engine. The result is shown in Table VI and the "√" represents the malicious HTTP requests detected by scanning engines. We analyze the HTTP malicious traffic proportion in the same family. As can be seen from the red part on Figure 6(a), we find that in different families, the malicious HTTP requests are quite different. This indicates that HTTP request can help identify the malware behavior. However, it is not so ideal to only use HTTP requests. Some malware families do not have any HTTP request, which means only by estimating the HTTP request to identify malware is not so effective. We analyze the proportion of malicious HTTP requests, and find that 40.89% of HTTP requests in our top 24 families are malicious.

But in fact, these scanning engines cannot detect all malicious requests. As shown in Figure 6(a) the green part, we analyze the proportion of malicious requests that are not recognized by scanning engines. Among these malwares that are not identified, some even account for a major part in the



(a) Packet number ratio of downlink to uplink traffic amount

(b) Byte size ratio of downlink to uplink traffic amount

Figure 5: The ratio of downlink to uplink traffic amount in top 24 families

same family. For example, in $Kmin$ family, HTTP requests $transit.5j5w.com, jujoy.5y3g.com, transit.zhiyule.com$ are both malicious (which accounts for 96.81%), which transfer the user's IMSI, phone number, SDK version to the server. The other one is $i.yangruiling.com$ in $FakeInstaller$(account for 4.12%), this site was found in 2012, it reads a series of encrypted characters from its file, then composes the control server address $i.yangruiling.com$ after decryption. Another one is $mlo6.com$ in $DroidDream$ (account for 6.45%), which is one of remote control servers. Therefore, the scanning results are not completely reliable according to our analysis.

In addition to analyze malicious HTTP requests, we also analyze the other sources of the requests. We find that a large part of the rest traffic are mobile Ad traffic, which account for 34.59%. The Ad traffic will be discussed in next section. Finally, for the rest HTTP request, including some identifiable source traffic [6], such as Google traffic, third-part traffic. Since they are out of the scope of this paper, we put them into the other source traffic as the "$*$" representation in Table VI.

4) Ad Traffic Analysis: In this section, we analyze the Ad traffic, which occupies a majority of all the HTTP traffic(34.59%). First, we use the scanning engines to exclude malware traffic. Then, we screen the Ad traffic according to the research about Ad libraries, which is used in [7]. According to the information from CrunchBase (crunchbase.com), we conduct IP address lookup, DNS and whois, additional information and knowledge from public databases to identify the type of traffic sources after resolving the top-level domains of the network traffic in [6]. Finally, we get the percentage of Ad traffic in the same family, which is shown in Figure 6(b). Compared with Figure 6(a), we discover an interesting phenomenon: if Ad traffic takes a larger percentage of the traffic, the identified malicious requests will have a smaller percentage of the traffic. For example, in the $DroidKungFu$ family, the Ad traffic percentage is 91.13%, but the malicious requests percentage is only 8.87%, in contrast to $FakeInstaller$, the Ad traffic percentage is 0.00%, but the malicious requests percentage reaches 89.69%. Because our samples are all malicious, It seems that the amounts of Ad traffic will lower the detection rate. Therefore, we think that these Ad traffic would cause damage to users stealthily.

There are some exceptions, such as $Geinimi$ and $Imlog$, both have 0.00% ad traffic and the malicious requests percentage are 8.93% and 14.29%. $Yzhc$ and $Jifake$(account for 1.19% of 5560) have few HTTP requests. Furthermore, after analyzing some Ad traffic sources in details, we find that many are really malicious, for example, $r.domob.cn$ in $GinMaster$, it used a method to collect user information, $waps.ysler.com$ in $Imlog$ has been hijacked to transfer to other sites. In addition, comparing 69.55% (use DNS query to identify malware behavior) to 40.89% (use HTTP request to identify malware behavior), plus the result that Ad traffic account for 34.59% in HTTP traffic, we can draw a conclusion that Ad traffic would affect the malware detection to a certain extent.

D. Malicious Network Feature Analysis

Based on the analysis above, we learn that the majority of the traffic traces on the application layer are DNS and HTTP traffic, which account for 10.24% and 89.62% in the top 24 families. Meanwhile, 69.55% DNS queries and 40.89% HTTP requests are malicious, malware traffic (40.90%) and Ad traffic (34.59%)are two major compositions of the HTTP traffic. At the same time, some features could help us understand the malware network behaviors better, and even help identify the malware by using features like DNS queries and HTTP requests, HTTP packet length, and the ratio of downlink to uplink traffic amount. Of course, these features are a part of many features which we will continue to discover in our future work.

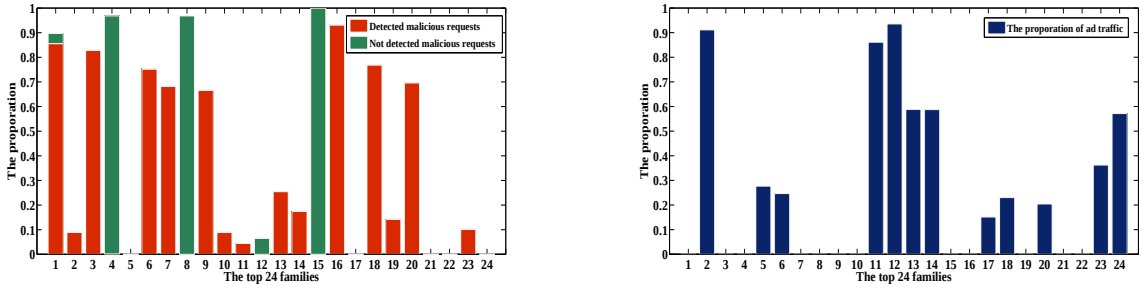
1) DNS Query and HTTP Request: Malicious DNS queries and HTTP requests could be used to set up a blacklist, showed in Table VII, we have the malicious URLs and their corresponding IP addresses.Notably, we also count the CNAME that corresponds to the same malicious IP. But since these CNAMEs take a small part, we do not list them here.

2) HTTP Packet Length: HTTP packet length includes the average length, maximum length and minimum length, meanwhile, based on the HTTP packet length distribution, we calculate its HTTP length mathematical expectation and variance, Then, we make a comparison with the benchmark values, and use their difference or ratio as a feature to identify a malware.

3) Ratio of Downlink to Uplink Traffic Amount: Based on the analysis about the ratio of downlink to uplink traffic

Table VI: An overview of the top 5 HTTP requests in the top 24 families; "$\surd$" represents the recognized malicious requests, "$\times$" represents the malicious requests not recognized by scanning engines, "$\star$" represents the ad traffic, "$*$" represents the other source traffic.

FamilyName	Top request 1	Result	Top request 2	Result	Top request 3	Result	Top request 4	Result	Top request 5	Result
FakeInstaller	91.213.175.176	$\surd$	waply.ru	$\surd$	rukodelniza.ru	$*$	91.213.175.148	$\surd$	i.yangruiling.com	$\times$
DroidKungFu	static.adwo.com	$\star$	www.adwo.com	$\star$	req.adsmogo.com	$\star$	app.wapx.cn	$\surd$	r2.adwo.com	$\star$
Plankton	ad.leadboltapps.net	$\surd$	api.airpush.com	$\surd$	apperhand.com	$\surd$	phoneliving.com	$*$	searchmobile.com	$\surd$
Opfake	m-001.net	$\times$	62.109.21.90	$\times$	lerpoa.pz9.ru	$\times$	91.211.88.65	$\times$	prowap.biz	$\surd$
GinMaster	www.google-analytics.com	$*$	ads.mobclix.com	$\star$	gw.youmi.net	$\star$	r.domob.cn	$\star$	r2.adwo.com	$\star$
BaseBridge	b4.cookier.co.cc:8080	$\surd$	appsrv1.madserving.cn	$\star$	dev.adtouchnetwork.net	$\surd$	b3.8866.org:8080	$\surd$	req.adsmogo.com	$\star$
Iconosys	smsreplier.net	$\surd$	blackflyday	$*$	184.154.161.66	$*$				
Kmin	transit.5j5w.com	$\times$	jujoy.5y3g.com	$\times$	transit.zhiyule.com	$\times$	crl.globalsign.com	$*$		
FakeDoc	bongacams.com	$\surd$	tools.bongacash.com	$*$	moba.rsigma.com	$*$	adserver.kimia.es	$\surd$	cn.bongacams.com	$\surd$
Geinimi	google.funimoe.com:8080	$*$	getyourchadon.com	$*$	www.google-analytics.com	$*$	test.adpooh.com:80	$\surd$	data.flurry.com	$*$
Adrd	gw.youmi.net	$\star$	ade.wooboo.com.cn	$\star$	222.186.14.13	$*$	adrd.taxuan.net	$\surd$	wap.casee.cn	$\star$
DroidDream	www.gstatic.com	$\star$	mlo6.com	$\times$	googleads.g.doubleclick.net	$\star$	api.admob.com	$\star$	mm.admob.com	$\star$
ExploitLinuxLotdoor	www.miidi.net	$\star$	test.adpooh.com:80	$\surd$	crl.globalsign.com	$*$	r2.adwo.com	$\star$	www.umeng.com	$\star$
Glodream	ade.wooboo.com.cn	$\star$	www.gstatic.com	$*$	lebar.gicp.net	$\surd$	report.adview.cn	$\star$	report.adview.cn	$\star$
MobileTx	img.tx.com.cn	$\times$	mobile.tx.com.cn	$\times$						
FakeRun	api.airpush.com	$\surd$	ad.leadboltapps.net	$\surd$	www.youtube.com	$*$	www.droidsettings.com	$*$		
SendPay	api.go108.cn	$*$	d.wiyun.com	$\star$						
Gappusin	app.wapx.cn	$\surd$	ads.wapx.cn	$\surd$	www.umeng.com	$\star$	r.domob.cn	$\star$	req.adsmogo.com	$\star$
Imlog	wp.ysler.com	$*$	www.imnet.us	$\surd$	waps.ysler.com	$*$				
SMSreg	ro.plus1.wapstart.ru	$\surd$	ro.plus1.wapstart.ru	$\star$	smartsms.org	$*$	www.google-analytics.com	$*$	www.apperhand.com	$\surd$
Yzhc	crl.globalsign.com	$*$								
Jifake	crl.globalsign.com	$*$								
Hamob	ads.appsgeyser.com	$\star$	www.tdvision.com	$*$	stat.appsgeyser.com	$*$	b.adinch.com	$\surd$	www.battleon.com	$*$
Boxer	media.admob.com	$\star$	www.gstatic.com	$\star$	googleads.g.doubleclick.net	$\star$	data.flurry.com	$*$	appstat.mmi.ru	$*$



(a) The proportion of malicious HTTP request in the top 24 families

(b) The proportion of ad traffic in the top 24 families

Figure 6: The proportion of the malicious and ad HTTP requests in top 24 families

amount, we know the packet number ratio would go along the red line (means the downlink equal to uplink), while the byte size ratio will be located above the red line.

4) Ad Traffic: Ad traffic account for most of all the generated traffic in some malware families, and our findings have shown that Ad would affect malware detection. For these malicious Ad servers which have been confirmed, we also put them into the blacklist.

In this paper, we have done an analysis about the traffic generated by the malware. In our future work, we will extend our work to compare with the legitimate traffic generated by the benign Apps.

IV. Related Work

The importance of mobile network is increasing with the proliferation of mobile devices, especially on the Android system. Previous studies have highlighted the weaknesses of the Android security model [8]. These researches are mostly based on static analysis, and open-source tools [9] are used to decompile and disassemble the source code. Recent studies focus on the malicious behaviors [10], these researches all analyze the malware dynamically at runtime. Meanwhile, several work have examined mobile device network traffic to learn about the their general network characteristics [11].

Network level analysis of malware behavior offers a complementary means of characterizing and mitigating malware. Cheng *et al.* [12] designed SmartSiren to collect the communication activity information from the smartphones. But it's impractical to run a agent on each smartphone. Tenenboim-Chekina *et al.* [13] described and analyzed a new type of mobile malware applications with self-updating capabilities, and then presented a network-based behavioral analysis for detecting such malware, their work has shown the network features between the malicious and the benign are much different, for instance, inbound and outbound bytes, inbound and outbound data, inner and outer time intervals. Furthermore, based on the applications' network traffic patterns, Shabtai *et al.* [14] designed a system that followed the hybrid Intrusion Detection Systems(IDS) approach and in the client-server architecture. But their work only focused on the several malicious types, which is not scalable.

Though there has been considerable efforts aiming to detect network malware and traffic analysis, there are not enough efforts spent on complete screening and systematic characterization of android malware network traffic through a large amount of samples.

Table VII: A blacklist set up by the top 3 DNS queries and the top 5 HTTP requests of the top 24 families

Malicious Domain Name	IP Address
waply.ru	78.140.131.119
i.yangruiling.com	108.61.11.3
app.wapx.cn	219.234.85.216,219.234.85.220,219.234.85.238, 219.234.85.222,219.234.85.240,219.234.85.243
api.airpush.com	67.222.106.169,67.222.111.118,67.222.111.117,162.219.250.210
ad.leadboltapps.net	54.243.235.3,54.243.236.68,23.23.255.172,54.243.206.52, 54.255.220.219,54.243.123.77,54.243.236.68,23.21.115.13
m-001.net	188.42.243.203
prowap.biz	185.53.178.20
dev.adtouchnetwork.net	109.201.133.191
b3.8866.org	117.21.224.222,111.74.238.109
b4.cookier.co.cc:8080	199.2.137.140
b3.8866.org:8080	117.21.224.222
smsreplier.net	50.56.218.189
bongacams.com	64.210.142.13
adserver.kimia.es	176.28.99.254,176.28.99.235
test.adpooh.com:80	115.238.144.30
log.meego91.comm	211.5.133.18
adrd.taxuan.net	69.195.129.70
ml06.com	74.63.220.83
lebar.gicp.net	174.128.255.228
mobile.tx.com.cn	221.181.68.19
ads.wapx.cn	219.234.85.237,219.234.85.236,219.234.85.214
www.imnet.us	72.52.4.121
ro.plus1.wapstart.ru	81.19.95.7,81.19.95.8
www.apperhand.com	211.5.133.18
api.tapfortap.com	198.61.246.5
b.adinch.com	192.175.102.93
c.vserv.mobi	46.137.90.66
gaga01.net	208.91.197.104
client.go360days.com	204.11.56.26
axy.waplove.cn	69.195.129.70
admin.51widgets.com	63.156.206.202
Null	91.213.175.176
Null	91.213.175.148
Null	62.109.21.90
Null	91.211.88.65
joinload.ru	Null
push.mobsqueeze.com	Null

V. Conclusion

In this paper, we provided an effective Android malware traffic generation and collection mechanism in the real Internet environment. At the same time, our work provided an in-depth analysis into the malware running and traffic generation procedure, and produced a traffic data set of 500.4MB size based on an Android malware dataset with 5560 samples. We will make the traffic dataset public to foster more research on the mobile malware's network behaviors.

We also presented a study of malware traffic features in the first few minutes, such as DNS query, HTTP packet length, ratio of downlink to uplink traffic amount, and HTTP requests. By analyzing the DNS queries and HTTP requests, we find that 69.55% and 40.89% of them are directed to malicious web sites. On the one hand, it confirms that a large part of malware would generate malicious behaviors in the first few minutes. Meanwhile, it validates that exploiting DNS query and HTTP request to identify the malware on the URLVoid could reach a minimum detection rate: 69.55% and 40.89%. Moreover, malware traffic (40.90%) and Ad traffic(34.59%) are two major compositions of the HTTP traffic. By analyzing the Ad traffic, we find that Ad traffic would affect malware detection.

Therefore, in future, we will leverage the Android malware traffic and reputation systems to research and develop a network traffic based anti-malware system at the network access point to identify emerging mobile threats.

Acknowledgment

This work was supported by the National Natural Science Foundation of China under Grants No.61472164,the Natural Science Foundation of Shandong Province under Grants No.ZR2014JL042 and No.ZR2012FM010.

References

[1] Y. Zhou and X. Jiang, "Dissecting android malware: Characterization and evolution," in *Security and Privacy (SP), 2012 IEEE Symposium on*, 2012, pp. 95–109.

[2] D. Arp, M. Spreitzenbarth, M. Hübner, H. Gascon, K. Rieck, and C. Siemens, "Drebin: Effective and explainable detection of android malware in your pocket," 2014.

[3] J. M. James Talbot, *Learning Android Application Programming: A Hands-On Guide to Building Android Applications*, 1st ed. Addison-Wesley Professional, 11 2014.

[4] G. C. Project, "Monkeyrunner," http://developer.android.com/guide/developing/tools/monkeyrunner_concepts.html.

[5] N. C. Srl, "Urlvoid," http://www.urlvoid.com/.

[6] X. Wei, L. Gomez, I. Neamtiu, and M. Faloutsos, "Profiledroid: multi-layer profiling of android applications," *Proceedings of Annual International Conference on Mobile Computing & Networking Mobicom*, vol. 11, no. 1, pp. 137–148, 2012.

[7] S. Dai, A. Tongaonkar, X. Wang, A. Nucci, and D. Song, "Networkprofiler: Towards automatic fingerprinting of android apps," in *INFOCOM, 2013 Proceedings IEEE*, 2013, pp. 809–817.

[8] Z. Fang, W. Han, and Y. Li, "Permission based android security: Issues and countermeasures," *Computers & Security*, vol. 43, no. 6, pp. 205–218, 2014.

[9] J. Freke, "Smali, an assembler/disassembler for android's dex format," *Google Project Hosting [online] http://code.google.com/p/smali*, 2013.

[10] L.-K. Yan and H. Yin, "Droidscope: Seamlessly reconstructing the os and dalvik semantic views for dynamic android malware analysis," in *USENIX Security Symposium*, 2012, Conference Proceedings, pp. 569–584.

[11] H. Falaki, D. Lymberopoulos, R. Mahajan, S. Kandula, and D. Estrin, "A first look at traffic on smartphones," *Wpi Edu*, 2010.

[12] J. Cheng, S. H. Y. Wong, H. Yang, and S. Lu, "Smartsiren: virus detection and alert for smartphones," *SmartSiren: virus detection and alert for smartphones*, pp. 258–71, 2007.

[13] L.Tenenboim-Chekina, A. O. Barad, D. M. Shabtai, B. Shapira, and Y. Elovici, "Detecting application update attack on mobile devices through network features." INFOCOM'2013, 2013.

[14] A. Shabtai, L. Tenenboim-Chekina, D. Mimran, L. Rokach, B. Shapira, and Y. Elovici, "Mobile malware detection through analysis of deviations in application network behavior," *Computers & Security*, vol. 43, no. 6, pp. 1–18, 2014.