Android Botnets: What URLs are Telling Us

Andi Fitriah Abdul Kadir(✉), Natalia Stakhanova, and Ali Akbar Ghorbani

Faculty of Computer Science, University of New Brunswick,
Fredericton, New Brunswick, Canada
{andi.fitriah,natalia.stakhanova,ghorbani}@unb.ca

Abstract. Botnets have traditionally been seen as a threat to personal computers; however, the recent shift to mobile platforms resulted in a wave of new botnets. Due to its popularity, Android mobile Operating System became the most targeted platform. In spite of rising numbers, there is a significant gap in understanding the nature of mobile botnets and their communication characteristics. In this paper, we address this gap and provide a deep analysis of Command and Control (C&C) and built-in URLs of Android botnets detected since the first appearance of the Android platform. By combining both static and dynamic analyses with visualization, we uncover the relationships between the majority of the analyzed botnet families and offer an insight into each malicious infrastructure. As a part of this study we compile and offer to the research community a dataset containing 1929 samples representing 14 Android botnet families.

Keywords: Android botnet · Malware · URL · Visualization

1 Introduction

The proliferation of mobile platforms in our daily lives has quickly brought mobile malware to the forefront of security concerns. Almost non-existent before the official release of the Android platform in 2008, nowadays mobile malware is a serious threat to modern mobile devices. Among them, mobile botnets are quickly gaining the attention of the research community. The recent report published by Sophos [2] noted the sophistication and highly stealthy nature of rapidly appearing mobile botnets.

Although the first studies in this domain only offered proof of concept models of mobile botnets, they identified the potential of the mobile platform for creating more sophisticated and stealthy botnets [26]. Indeed, the resource-constraint environment of smartphones, which are unable to afford computationally intensive operations presents significant challenges to the development of solutions for their detection. Botnets in the mobile environment have to be resource-aware to continue their operation and remain undetected. This effectively forces mobile botnets to curtail their communication to a minimum and leverage alternative hard-to-detect channels (e.g., audio/video sensors [18], SMS/MMS [16,27,19,26]).

M. Qiu et al. (Eds.): NSS 2015, LNCS 9408, pp. 78–91, 2015.
DOI: 10.1007/978-3-319-25645-0_6

Mobility also opens up new avenues for old attacks. Mobile phones are a rich source of sensitive information, traditionally not available to stationary computers (e.g., location information, user's activities). This creates an opportunity for new context-aware mobile botnets to be able to exfiltrate information typically not monitored by traditional detection systems. General detection of mobile malware and in particular detection of mobile botnets have been extensively studied in the last several years [25,14,15,11,10]. Aiming to address specific features of the existing botnets, these studies appear isolated and give patchy solutions to an ever growing problem. This disparity stems from a lack of solid understanding of modern mobile botnets functionality and specifics differentiating them from their traditional counterparts. In this work we aim to address this gap and offer an insight into the most popular mobile botnet families. We analyze 14 Android botnet families detected in the wild since 2010. Given an unprecedented growth of Android malware and botnets (98% of all mobile malware [13]), in this study we focus primarily on the Android platform. Our goal is to provide a deep understanding of Android botnets' characteristics that will facilitate the development of advanced mobile botnet detection approaches.

The contribution of our work is three-fold. First, we conduct a thorough investigation of the Android botnet families, their characteristics and communication behaviour. Second, through static and dynamic analysis, we extract and visualize all embedded URLs, including the obfuscated URLs. During our analysis, we identify hidden encryption keys stored within the samples that allow us to reveal previously unknown features of Android botnets which are currently used to avoid detection. This analysis not only allows us to demonstrate the relationships between botnet families, but also helps us to illustrate the C&C communication patterns. Finally, we release the accumulated dataset containing 1929 botnet samples to the research community. To the best of our knowledge, this study is the first of its kind that offers a thorough analysis of mobile botnets' URLs on the Android platform.

The rest of the paper is organized as follows: Section 2 presents the related work on Android botnets and Section 3 discusses the collected dataset. Section 4 presents the approach we used to analyze the URLs of the Android botnets. Section 5 presents the discovered trends. Finally, Section 6 concludes the paper with some remarks about the implication of the work.

2 Related Work

With rapid advancement of botnets and mobile device security quickly becoming an urgent necessity, researchers have focused their attention on this problem. One of the first attempts to illustrate the potential impact of a small-size mobile botnet was offered by Traynor et al. [24]. It was quickly followed by a series of studies introducing more advanced and powerful mobile botnet designs capable of significant damage: Andbot, a botnet exploiting URL fluxstrategy [26], Android botnet based on Google's Cloud to Device Messaging (C2DM) service [28], Android botnet leveraging out-of-bound communication channels

(i.e., audio, ambient light, and magnetic field) [18], mobile multi-targeted botnet [17]. Realization of mobile botnet attacks' potential triggered the development of defence techniques, many of which specifically targeted individual characteristics of botnets [23,14,11,16,27,19,10,25,15].

We provide these approaches with an understanding of the Android botnet communication characteristics and thus provide tools to improve the effectiveness of botnet analysis. One of the techniques we employ in our study is visualization. Visualization has been seen as beneficial in many domains including mobile security. One of the first works in this area was offered by Barrera et al. [12]. The authors employed visualization to observe and analyze permission usage in malicious Android applications. This work was followed by Luoshi et al. [22] where the authors used the Gephi visualization tool for detecting Android malware by investigating the relationships between Android function calls and their paths. The recent work by Hosseinkhani et al. [21] introduced Papilio, a new visualization technique for illustrating real-world Android application permissions. In the context of our study, we adopt visualization to investigate URLs of Android botnets.

3 Dataset

To provide a comprehensive evaluation of Android botnets, we gathered a large collection of Android botnet samples representing 14 botnet families. These families represent early and mature versions of Android botnets that are chosen primarily due to their popularity. The summary of these families characteristics are provided in Table 1. Our accumulated dataset combines botnet samples from the Android Genome Malware project [30], malware security blog [1], VirusTotal [3] and samples provided by a well-known anti-malware vendor. Overall, our dataset includes 1929 samples of Android application package (APK) spanning a period from 2010 (the first appearance of Android botnet) to 2014. This dataset covers a large number of existing Android botnets, which reflects the current status of Android Malware. Figure 1 illustrates the cumulative growth of samples in our dataset. We have noticed that even though the first botnet was discovered in 2010, there are few samples that have been created earlier (this is witnessed by the creation dates on the ZIP files) than the official discovery date in 2008, as depicted in Figure 1. This indicates that these botnets were unknown or unlabelled until they were discoved in 2010. We have released the accumulated dataset to the research community at http://www.unb.ca/research/iscx/dataset/index.html

4 Extracting URLs

Traditionally, URLs are either contained in the file meta-data (e.g., the application links for updates) or embedded in malware code as plain text or obfuscated strings. URLs from meta-data are easily extracted with a simple regular

Table 1. Android botnet characteristics

Botnet Family	Year	Market Origin	C&C Type	Target Country	Backdoor	Data Theft	Drive-by Download	Exploit technique	Infected SMS	Mobile Banking Attack	Ransomware	Repackaged Application	Social Engineering	Trojanised Application
					Propagation and Attack Types									
Wroba	2014	3rd-party	SMS/HTTP	Korea					✓	✓				✓
Pletor	2014	3rd-party	SMS/HTTP								✓			✓
Sandroid	2014	3rd-party	SMS	MiddleEast						✓	✓			✓
NotCompatible	2014	forged site	HTTP				✓	✓						
MisoSMS	2013	3rd-party	Email	Korea	✓			✓						✓
Bmaster	2012	3rd-party	HTTP	China	✓			✓				✓		
RootSmart	2012	3rd-party	HTTP	China	✓			✓				✓		
TigerBot	2012	3rd-party	SMS		✓	✓								✓
AnserverBot	2011	official	HTTP		✓				✓				✓	
DroidDream	2011	official	HTTP		✓	✓	✓					✓		✓
NickySpy	2011	3rd-party	SMS		✓							✓		
PJapps	2011	3rd-party	HTTP									✓		✓
Geinimi	2010	3rd-party	HTTP	China	✓	✓						✓		
Zitmo	2010	3rd-party	SMS	Europe					✓	✓		✓	✓	

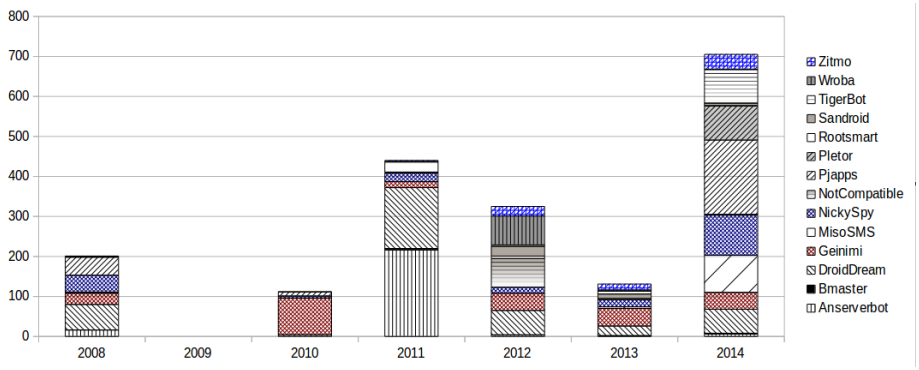

Fig. 1. The yearly breakdown of the collected Android botnet families

expression-based search. We refer to these extracted meta-data URLs as built-in URLs. To obtain URLs, we employ both static and dynamic analyses. The combination of static and dynamic analyses is necessary here as many botnets

Table 2. Overview of the extracted URLs

Botnet Family	Total samples	Static analysis(URL)		Dynamic analysis(URL)		Similar URL
		Total	Unique	Total	Unique	
Anserverbot	244	903	130	289 974	115	21
Bmaster	6	148	69	4480	43	7
DroidDream	363	6 451	850	4 443 219	502	21
Geinimi	264	1 548	406	342 760	181	11
MisoSMS	100	450	60	179 907	224	9
NickySpy	199	5 402	411	34 303	171	24
NotCompatible	76	72 200	24	10 875	5	0
PJapps	244	4 783	676	182 932	200	13
Pletor	85	307	37	17 181	12	8
Rootsmart	28	486	16	64	14	0
Sandroid	44	1 305	218	2 566	179	11
TigerBot	96	555	47	6 188	35	4
Wroba	100	2 372	31	NA	NA	NA
Zitmo	80	7 648	74	158 508	138	26
Total	1 929	104 558	3 049	5 672 957	1 819	155

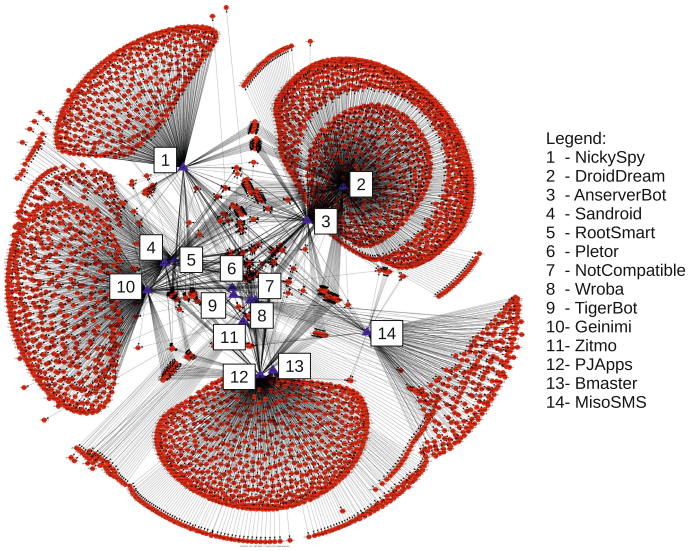

Fig. 2. Relationships between 4 713 URLs extracted from the botnet dataset

(especially with advanced capabilities) will hide URLs from static analysis. A sophisticated botnet would hide its malicious intent and avoid putting C&C URLs as meta-data. Dynamic analysis, on the other hand, forces a botnet sample to reveal these hidden URLs.

In the static analysis, we have customized scripts that leverage regular expressions and keywords such as *http, password, key, DES*. We looked at the similar keyword pattern between each family. For instance, searching the *const-string*

keyword can give us more results of URLs than the plain *http* keyword. By examining the existing pattern for each botnet family, we managed to extract even the encoded URLs. This is obtained by using binary code searching. Searching binary code, however, requires disassembly of APK file and analysis of *.dex* file bytecode. Using the *baksmali* disassembler, we retrieved bytecode from each sample and string portion of its data section. We referred to domains and IP addresses extracted from the data section part of the code as C&C URLs. To confirm the discovery of the C&C URLs, we followed several patterns from the existing reports such as the use of port 8080 in their communications (which can be clearly seen in the URLs).

The dynamic analysis was conducted using Anubis [9], a web-based malware analysis tool. Anubis provides both static and dynamic analysis reports which cover the following aspects of APK files: activities, services, broadcast receivers, required permissions, used permissions, features, URLs, file operations, network operations, crypto operations, started services, and native libraries loaded. In this research, we focus primarily on the analysis of the URLs. We have extracted all the URLs from the collected xml files generated by Anubis except for the Wroba family (due to the Anubis startup-dependency issues). Following the described procedure, we extracted over 5 million of URLs (5, 777, 515) from our collected dataset (see Table 2). This resulted in 4 868 unique URLs; 155 of them are overlapping, which reduces the overall unique URLs to 4 713. Among them, we discovered 47 are the C&C URLs and the remaining 4 666 are the built-in URLs. Visualizing this information gives a high-level understanding of the relationships between botnet families. This is clearly seen in Figure 2 that displays all the unique URLs extracted from the 14 botnet families. As the visualization shows, there is a significant sharing of URL resources between seemingly isolated botnet families.

5 Analyzing URL Patterns

5.1 Resource Sharing among Botnet Families

One interesting aspect of the analyzed botnet families is the sharing of resources, i.e., not only the same URLs are being reused within and across families, but also encryption keys employed to obfuscate these addresses. Moreover, they are also sharing the same nameservers as shown in Figure 3.

Built-in vs C&C URLs. The sharing of URLs is clearly visible in Venn diagrams illustrating the relationships between the built-in and C&C URLs. While there is no overlap between C&C and built-in URLs, there is a clear reuse within each category. For instance, as Figure 4 shows, all the families are linked to each other through a significant reuse of built-in URLs. This might be an indication of legitimate resources being a main vehicle of botnet malware. This is also confirmed by the scanning results, as only a minor portion of these built-in URLs are malicious. It should be noted that the same pattern exists between the built-in and C&C URLs, which shows that the DroidDream is a subset of

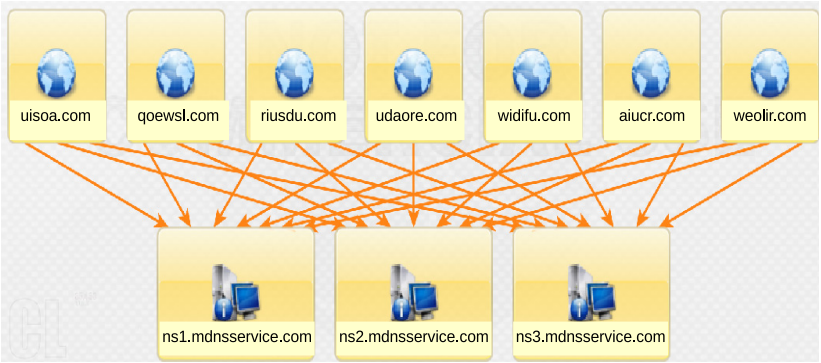

Fig. 3. Examples of nameservers sharing of C&C domains for the Geinimi Family

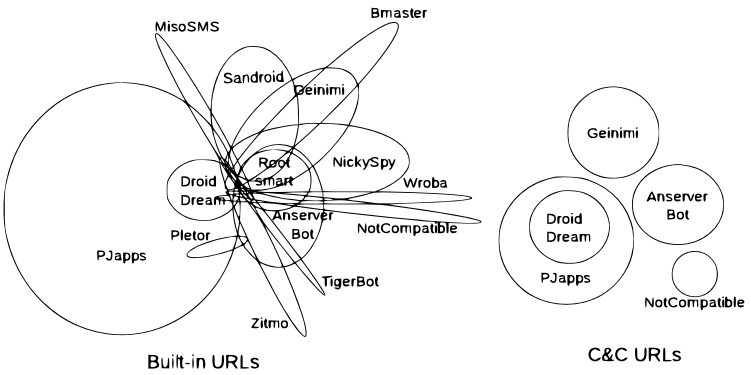

Fig. 4. Built-in and C&C URL relationships

the PJapps family. Based on this finding, we can infer that even though there is no URL similarity between the built-in and C&C URLs, they employ the same pattern, which might be an indication that the DroidDream family is actually the evolution of the PJapps family.

Interfamily Relationships. We observed the following types of relationships between employed addresses within the same family:

- one-to-one: when an APK file is associated with a single URL.
- one-to-many: when an APK file contains many URLs
- many-to-many: when many APK files are associated with many URLs.

Figure 5 illustrates these types of C&C URL-relationships. For instance, PJapps adopted a one-to-one relationship where it uses a single URL to pull down a command. Geinimi botnet uses a one-to-many relationship to make several attempts to connect to multiple C&C servers (up to 10 distinct URLs).

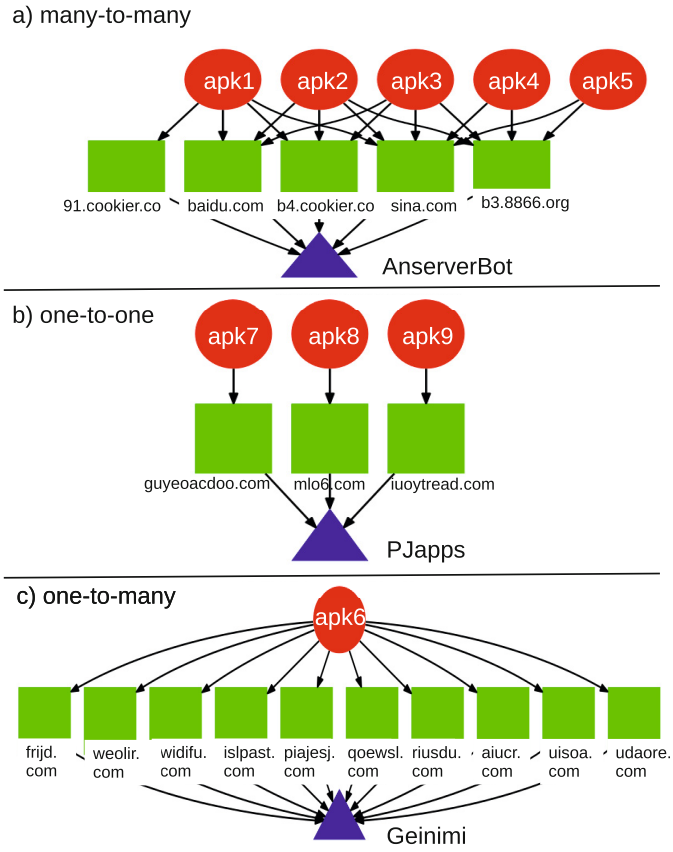

Fig. 5. Types of C&C URL-relationships discovered

AnserverBot, on the other hand, employed the many-to-many relationship where various APK files used the same set of URLs (up to 3 distinct URLs) to communicate with C&C servers. However, we also noticed that some families such as PJapps and DroidDream adopted multiple relationships in communicating with their C&C servers.

Furthermore, there is also a significant relationship between the C&C URLs and their encryption pattern within the same family. For instance, all of 365 APK files in the DroidDream family adopted the same technique for encrypting the C&C URLs. This botnet family stored the hardcoded URLs in the same folder and defined the key to decrypt these URLs in the Java code with the keyword of *PASSWORD_CRYPT_KEY*. A previous report by McAfee in 2011 [4] discovered that DroidDream variants save the encrypted configuration using the file name *prefer.dat* in the Asset folder. The samples we analyzed use *sense.tcd* and *small.use* to store the same DES decryption/encryption key as before -

DDH#X%L. This indicates that the botmasters have changed the configuration file without changing the encryption pattern (file path and key). Specifically, we discovered that 253 APK files stored their configuration in *sense.tcd* file, 148 samples saved their encrypted configuration in *prefer.dat* file, and only 4 samples used the *small.use* file. All of these 405 APK files linked to the same set of URLs (up to 7 distinct URLs) and adopted the many-to-many URL relationship. This demonstrates that botnets commonly recycle their resources.

Intrafamily Relationships. In our analysis we also discovered a significant relationship between the C&C URLs from different families. By searching and visualizing the encryption, we found 101 APK files of the PJapps family applied the same encryption technique as DroidDream, 88 of them stored their encryption configuration in the *prefer.dat* and another 13 APK files used the *sense.tcd* file.Both used the same DES decryption/encryption key as before, *DDH#X%L*. To further investigate, we checked these APKs with VirusTotal to see the results of its detection. The results are diverse as different APKs showed a different detection ratio. However, we noticed that some vendors such as Sophos consistently detect these APKs as PJapps, while other vendors such as F-secure declared these APKs as DroidDream. As both families appeared in the same year, we infer that this is an example of malware evolution where the botmasters replicate the APK file and repackage with improved techniques to evade detection. By scrutinizing the newest samples of DroidDream and PJapps, we found a new pair of DES decryption/encryption keys (*pG3N0̃8f?* and *G#R%AP̂H?*) which are different from the previously seen keys. None of these keys matches the encrypted files (*prefer.dat, small.use, sense.tcd*). We suspect that this is another improvement employed by the bot master in the newer versions of malware.

Moreover, there is also a significant relationship between the C&C URLs and their proxy information within the same family. By visualizing the proxy numbers, we noted that the same set of URLs shares the same proxy numbers. For example, all of the 7-set C&C URLs in a DroidDream family use the same proxy numbers in uploading (proxy number 7) and feeding (proxy number 9) the C&C server. On the other hand, a different 2-set of URLs is also linked to another proxy (proxy number 2). Specifically, this proxy information comes together with the C&C URLs that we have decrypted using DES encryption.

5.2 C&C Addresses Obfuscation

One of the techniques employed by Android malware to prevent detection and the subsequent analysis is the obfuscation of C&C servers' addresses. There have been a number of methods reported in the past (e.g., plain text Java file in the DroidKungFu family [30]). Here we offer an overview of the techniques we observed in Android botnets.

Encryption. Note that as well as the AnserverBot family that used encoding scheme to dodge detection as previously reported by Zhou and Jiang [30], other

Table 3. Overview of the employed encryption techniques

Botnet Family	Total Encrypted URL	Encryption types				
		Rot-10	XOR	DES	AES	Base64
AnserverBot	141	2	0	32	99	8
Bmaster	11	0	7	0	4	0
DroidDream	104	9	1	11	78	5
Geinimi	46	4	0	10	29	3
MisoSMS	2	0	0	0	1	1
NotCompatible	145	0	0	32	113	0
Nickispy	4	0	0	0	1	3
PJapps	27	11	0	7	1	8
Pletor	81	0	0	0	81	0
Rootsmart	25	0	0	0	25	0
Sandroid	2	0	0	0	9	2
TigerBot	23	0	0	0	0	23
Wroba	3	0	0	0	3	0
Zitmo	11	0	0	3	8	0
Total	625	26	8	95	452	53

families such as Geinimi, PJapps, and DroidDream have also adopted an encryption algorithm in their APK Botnet. A summary of the encrypted URLs and their corresponding encoding algorithms is given in Table 3.

In essence, AnserverBot adapts the popular Base64 scheme with a custom index table. PJapps customized this encoding scheme, employing Base64 with a pattern of skipping every other letter in strings. DroidDream on the other hand, employed both encoding and encrypting techniques such as XOR cipher, ROT-10 cipher, Advanced Encryption Standard (AES), and Data Encryption Standard (DES) algorithm provided with its three different keys of encryption: *DDH#X%LT?, pG3N0̃8f?, and G#R%AP̂H?.*

Exploiting DNS. Another interesting finding of C&C URL pattern is on the Domain Name System (DNS). We found that the C&C URLs exploit its DNS by adopting the Domain Generation Algorithm (DGA) and the URL obfuscation techniques. Table 4 lists 3 types of URL obfuscation techniques commonly used by attackers [20]. Accordingly, the C&C URLs in the collected dataset used the following types: Type I - obfuscating the host with an IP address, Type III - obfuscating with the large hostname, and, Type IV - unknown or misspelled domain. However, the behavior of Type II - obfuscating the host with another domain is not found in our dataset. In a similar way, the C&C URLs are used the DGA technique as rendezvous points with their C&C servers. As such, out of 47 C&C URLs that we have extracted, 33 of them have employed the DGA. For example in the DroidDream family, the domain names of the C&C URLs (http://ju5o.com/zpmq.jsp,http://mlo6.com/owxnf.jsp,http://ya3k.com/bksy.jsp) contain both random characters and numbers, which indicates the use of DGA.

Table 4. Commonly used URL obfuscation techniques

TYPE	Descriptive examples	Family
I	http://184.105.245.17:8080/GMServer/GMServlet&PJapps	
III	http://91.cookier.co.cc:8080/jk_center/91/ad.xml	AnserverBot
IV	www.qoewsl.com:8080;	Geimini

Utilizing Public Blog. A previous technical report of AnserverBot in 2011 [29] claimed that Anserverbot was the first one in Android Malware history that used public blogs as its C&C servers to deliver commands to bot clients. According to the authors, if the connection to the C&C server is not successful, the botnet will start connecting to the public blog for the updated C&C server and then use this as a new C&C server. Moreover, the information about new C&C servers is being published on a public blog (as encrypted postings) and the C&C URLs are hard-coded using Base64 scheme (with a custom index table). Based on this information, we employed customized scripts that leverage regular expressions to search for C&C URLs. Through this search, we extracted 830 C&C URLs (8 unique URLs) from the 243 APKs of the AnserverBot family that exploit public blogs. As Figure 5 shows, all these APKs have adopted many-to-many relationships. For instance, both of these URLs from *baidu.com* and *91.cookier.co.cc* link to the same 175 APKs. This shows that the AnserverBot has a limited number of public blogs to be used as C&C servers.

5.3 Detection of Botnet Samples

All of the analyzed samples were collected from various sources and initially labeled as botnet malware. To measure the effectiveness of modern anti-virus scanners we analyzed the collected samples using the VirusTotal service [3]. Since the VirusTotal service incorporates a large selection of anti-virus scanners, which use different detection strategies, we identify the number of scanners that detect our samples as botnet, together with their average detection rate. The results are encouraging as the majority of our samples were recognized correctly with a few exceptions. For example, a low detection rate, 86%, was obtained for Sandroid family that appeared fairly recently (in 2014). All the older families, for the most part, were detected. Note that none of the families were recognized by all existing anti-virus solutions.

We also looked at the maliciousness of the extracted URLs. Out of the 4 713 URLs (built-in and C&C) we extracted, 516 of them were detected as malicious: 27 out of 47 C&C URLs and 489 out of 4 666 built-in URLs. Surprisingly, some benign domains such as *maps.google, news.google*, and *Androids-market* were detected as malicious when provided as complete URLs. As expected, not all the C&C URLs considered as malicious are up-to-date. Considering the first appearance of some of these botnets (2010), we assume that the domain names might be reused by someone else for legitimate purposes. In this study, we are not conducting further analysis on these particular C&C domain names. However, if we look into the domain-level, most of the non-malicious URLs have

adopted the DGA technique in their DNS. To further illustrate the malicious URL relationships, we visualized these URLs based on their families.

We also cross-checked the extracted URLs with several well-known blacklists [5,8,7,6] as listed in Table 5. Interestingly, we found no matches with the extracted URLs. Even though many of the analyzed families are date back to 2010 and 2011, these botnet URLs are still not a part of these blacklists. Another point to make here is the reuse of the URLs; as our analysis shows, although Android botnets share many URLs, all of them are different from those used by other malware.

Table 5. Cross-check with blacklists

Name	Total URL/domain	Total URL Detected
Malware Domain Blocklist	24 070	0
Shalla Blacklist	179 593	0
URL Blacklist	242 548	0
Zeus Tracker	785 187	0
Total	1 231 398	0

6 Conclusion

In this work, we have looked at improved methods of Android botnet behavioural analysis based on URL analysis. We have also shown the major differences between Android botnet URLs and the benign ones and showed their relationship with blacklists and anti-virus scanners. We have discovered that Android botnets tend to encrypt various types of data including the URLs of C&C servers, the method names to be invoked, the file path of the payloads, and even the content of the payloads to prevent them from being reverse engineered. We confirmed that the mobile botnets are evolving and becoming more sophisticated; thus, the samples from 2012 till the recent ones are more dynamic. Most of the URLs of these samples are hard coded in the Malware but created on the fly. However, by focusing on the string pattern extraction and visualization through static and dynamic analysis, we have managed to extract and decode these URLs. Through this study, we were able to identify the variety of encryption techniques used by bot masters. This is achieved by extracting the strings and visualizing each apk file from its botnet family as well as the mappings between the URLs.

Acknowledgments. This research has been partially supported by the New Brunswick Innovation Foundation under research grant RAI 2015-090. The authors also gratefully acknowledge funding from International Islamic University Malaysia (IIUM) and would like to thank Alip Aswalip for sharing the malware samples.

References

1. Mobile malware mini dump. http://contagiominidump.blogspot.ca/ (accessed April 1, 2015)
2. Security threat trends. https://www.sophos.com/en-us/threat-center/medialibrary/PDFs/other/sophos-trends-and-predictions-2015.pdf (accessed August 1, 2015)
3. Virus total. https://www.virustotal.com/en/ (accessed June 12, 2015)
4. Android malware: Past, present, and future. http://www.locked.com/sites/default/files/android-malware-past-present-future-wp.pdf (accessed June 7, 2015)
5. Malware Blocklist. http://www.malwaredomains.com/ (accessed March 5, 2015)
6. Shalla's blacklists. http://www.shallalist.de/ (accessed March 5, 2015)
7. Url blacklist. http://www.urlblacklist.com/ (accessed March 5, 2015)
8. Zeus tracker. https://zeustracker.abuse.ch/blocklist.php (accessed March 5, 2015)
9. Anubis: web-based malware analysis for unknown binaries. https://anubis.iseclab.org/ (accessed May 30, 2015)
10. Abdelrahman, O.H., Gelenbe, E., Görbil, G., Oklander, B.: Mobile network anomaly detection and mitigation: the NEMESYS approach. In: Information Sciences and Systems 2013, pp. 429–438. Springer (2013)
11. Alzahrani, A.J., Ghorbani, A.A.: SMS mobile botnet detection using a multi-agent system: research in progress. In: Proceedings of the 1st International Workshop on Agents and CyberSecurity, pp. 2:1–2:8. ACM, New York (2014)
12. Barrera, D., Kayacik, H.G., van Oorschot, P.C., Somayaji, A.: A methodology for empirical analysis of permission-based security models and its application to Android. In: Proceedings of the 17th ACM Conference on Computer and Communications Security, pp. 73–84. ACM, New York (2010)
13. Chebyshev, V., Unuchek, R.: Mobile malware evolution. https://securelist.com/analysis/kaspersky-security-bulletin/58335/mobile-malware-evolution-2013/ (accessed March 5, 2015)
14. Choi, B., Choi, S.K., Cho, K.: Detection of mobile botnet using VPN. In: Proceedings of the 2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, pp. 142–148. IEEE Computer Society, Washington (2013)
15. Feizollah, A., Anuar, N.B., Salleh, R., Amalina, F., Maarof, R.R., Shamshirband, S.: A study of machine learning classifiers for anomaly-based mobile botnet detection. Malaysian Journal of Computer Science **26**(4) (2014)
16. Geng, G., Xu, G., Zhang, M., Yang, Y., Yang, G.: An improved SMS based heterogeneous mobile botnet model. In: 2011 IEEE International Conference on Information and Automation (ICIA), pp. 198–202, June 2011
17. Hamon, V.: Android botnets for multi-targeted attacks. Journal of Computer Virology and Hacking Techniques, 1–10 (2014)
18. Hasan, R., Saxena, N., Haleviz, T., Zawoad, S., Rinehart, D.: Sensing-enabled channels for hard-to-detect command and control of mobile devices. In: Proceedings of the 8th ACM SIGSAC Symposium on Information. Computer and Communications Security, pp. 469–480. ACM, New York (2013)
19. Hua, J., Sakurai, K.: A SMS-based mobile botnet using flooding algorithm. In: Ardagna, C.A., Zhou, J. (eds.) WISTP 2011. LNCS, vol. 6633, pp. 264–279. Springer, Heidelberg (2011)
20. Le, A., Markopoulou, A., Faloutsos, M.: Phishdef:url names say it all. In: 2011 Proceedings IEEE INFOCOM, pp. 191–195. IEEE (2011)

21. Loorak, M.H., Fong, P.W.L., Carpendale, S.: Papilio: Visualizing Android Application Permissions. Computer Graphics Forum **33**(3), 391–400 (2014). http://diglib.eg.org/EG/CGF/volume33/issue3/v33i3pp391-400.pdf
22. Luoshi, Z., Yan, N., Xiao, W., Zhaoguo, W., Yibo, X.: A3: automatic analysis of android malware. In: 1st International Workshop on Cloud Computing and Information Security. Atlantis Press (2013)
23. Pieterse, H., Olivier, M.: Android botnets on the rise: trends and characteristics. In: Information Security for South Africa (ISSA), pp. 1–5, August 2012
24. Traynor, P., Lin, M., Ongtang, M., Rao, V., Jaeger, T., McDaniel, P., La Porta, T.: On cellular botnets: measuring the impact of malicious devices on a cellular network core. In: Proceedings of the 16th ACM Conference on Computer and Communications Security, pp. 223–234. ACM, New York (2009)
25. Vural, I., Venter, H.: Mobile botnet detection using network forensics. In: Berre, A.J., Gómez-Pérez, A., Tutschku, K., Fensel, D. (eds.) FIS 2010. LNCS, vol. 6369, pp. 57–67. Springer, Heidelberg (2010)
26. Xiang, C., Binxing, F., Lihua, Y., Xiaoyi, L., Tianning, Z.: Andbot: towards advanced mobile botnets. In: Proceedings of the 4th USENIX Conference on Large-scale Exploits and Emergent Threats, p. 11. USENIX Association, Berkeley (2011)
27. Zeng, Y., Shin, K.G., Hu, X.: Design of SMS commanded-and-controlled and p2p-structured mobile botnets. In: Proceedings of the Fifth ACM Conference on Security and Privacy in Wireless and Mobile Networks, pp. 137–148. ACM, New York (2012)
28. Zhao, S., Lee, P.P.C., Lui, J.C.S., Guan, X., Ma, X., Tao, J.: Cloud-based push-styled mobile botnets: a case study of exploiting the cloud to device messaging service. In: Proceedings of the 28th Annual Computer Security Applications Conference, pp. 119–128. ACM, New York (2012)
29. Zhou, Y., Jiang, X.: An analysis of the Anserverbot trojan. Tech. rep., Technical report, NQ Mobile Security Research Center (2011)
30. Zhou, Y., Jiang, X.: Dissecting android malware: characterization and evolution. In: 2012 IEEE Symposium on Security and Privacy (SP), pp. 95–109, May 2012