UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

Behavioral Malware Detection Approaches for Android

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

DOCTOR OF PHILOSOPHY

By

Mehedee Zaman, Muhammad Rakib Amin, Tazrian Siddiqui
Norman, Oklahoma

2015

Behavioral Malware Detection Approaches for Android

A DISSERTATION APPROVED FOR THE
Department of Computer Science and Engineering

BY

Mehedee Zaman

Muhammad Rakib Amin

Tazrian Siddiqui

Dr. Shohrab Hossain

Dedication

This dissertation is dedicated to .

Acknowledgements

This work has been possible because of a number of individuals. First of all, I would like to express my sincere gratitude to my faculty advisor Dr. Mohammed Atiquzzaman for his expert advice, patience and continuous supervision throughout my PhD studies, which made this work possible.

I am truly grateful to my committee members: Dr. John K. Antonio, Dr. Changwook Kim, Dr. Dean Hougen and Dr. Zahed Siddique for their valuable advice and time to review the dissertation. Their comments and suggestions have been very helpful to improve the quality of this dissertation.

I am also thankful to all the faculty and staff members of the School of Computer Science at the University of Oklahoma for providing a supportive environment for my study. Moreover, numerous discussion with my fellow researchers Dr. Abu Zafar Shahriar, Mr. Sazzadur Rahman, Syed Maruful Huq and Yasmin Jahir, and their technical expertise led to the improvements of this work.

The author would like to acknowledge the financial support of National Aeronautics and Space Administration (NASA) to carry out this project.

Last but not least, I would like to thank my wife Nusrat and my daughter Tasneem for accompanying and supporting me through all the tough times, and my parents and sister for providing moral support.

Table of Contents

List of Tables

List of Figures

Abstract

Today mobile computing has become a necessity and we are witnessing explosive growth in the number of mobile devices accessing the Internet. To facilitate continuous Internet connectivity for nodes and networks in motion, mobility protocols are required and they exchange various signaling messages with the mobility infrastructure for protocol operation. Proliferation in mobile computing has raised several research issues for the mobility protocols. First, it is essential to perform cost and scalability analysis of mobility protocols to find out their resource requirement to cope with future expansion. Secondly, mobility protocols have survivability issues and are vulnerable to security threats, since wireless communication media can be easily accessible to intruders. The third challenge in mobile computing is the protection of signaling messages against losses due to high bandwidth requirement of multimedia in mobile environments. However, there is lack of existing works that focus on the quantitative analysis of cost, scalability, survivability and security of mobility protocols.

In this dissertation, we have performed comprehensive evaluation of mobility protocols. We have presented tools and methodologies required for the cost, scalability, survivability and security analysis of mobility protocols. We have proposed a dynamic scheduling algorithm to protect mobility signaling message against losses due to increased multimedia traffic in mobile environments and have also proposed a mobile network architecture that aims at maximizing bandwidth utilization. The analysis presented in this work can help network engineers compare different mobility protocols quantitatively, thereby choose one that is reliable, secure, survivable and scalable.

Chapter 1

Introduction

Next generation networks are gradually converging towards the all-IP networks which can enable true global mobility and Internet connectivity to mobile devices.

1.1 Introduction

Internet Protocol (IP) is the underlying communication protocol that allows an end host to get connected to other hosts over the public Internet. Therefore, to facilitate continuous Internet connectivity for mobile nodes, Internet Engineering Task Force (IETF) proposed Mobile IPv6 [1], an IP-based mobility protocol.

This aggregated mobility management can significantly reduce signaling requirement and power consumption.

1.2 Motivation and Problem Statement

In a mobile computing environment, a number of *network parameters* (such as, network size, mobility rate, traffic rate) influence the signaling costs related to mobility management. With the rapid growth and popularity of mobile and wireless networks,

Finally, mobility protocols can be vulnerable to security threats.

1.3 Objectives

The *objectives* of this research are as follows:

- The first objective of this research is to perform a comprehensive cost and scalability evaluation of the
- The second objective of this research is the quantitative evaluation of survivability of the mobility infrastructure and the associated components.
- The fourth objective of this research is to protect mobility protocols from security threats.
- Finally, mobility protocols require a realistic mobility model that can mimic the movement pattern of nodes in motion.

1.4 Contributions

The *contributions* of the dissertation are summarized as follows:

- Perform entity-wise cost evaluation of host and network mobility protocols.
- Perform quantitative scalability analysis of host and network mobility protocols.
- Perform multi-class queuing analysis and propose a dynamic scheduling algorithm to protect crucial control messages (of mobility management) against losses.

1.5 Organization of the Dissertation

The rest of the dissertation is organized as follows. Chapter **??** presents a review of host and network mobility protocols.

Chapter 2

Detection using network traffic analysis

In this chapter, we will discuss about malware detection using Network traffic analysis. This chapter contains 2 sections. Section 2.1 describes the outline of our strategy, and in section 2.2, we describe the procedure in details.

2.1 Strategy of Malware Detection

At first, we created log of URLs that are contacted by applications for a specific period of time. Then we tried to match each entry (URL) of the log with a list of known malicious domains. If a match is found, the application that contacted the malicious domain is a malware itself or has been affected by one.

2.1.1 Creating the App-URL table

App-URL table is a history/log of all attempts made by all applications to communicate with remote servers over HTTP. The table consists of **(url, app)** entries. Each HTTP request maps to a single entry, where **url** is the URL which is contacted, and **app** is the application that originated the HTTP request.

This process is further subdivided into four tasks:

2.1.1.1 Packet dumping

We have recorded all incoming and outgoing network packets to/from the android device for specific duration of time. This creates a packet dump file that contains information of which port number (of the mobile device) is accessing which URL.

2.1.1.2 Netstat Logging

To relate port numbers with applications, we periodically executed *netstat* [**?**] command throughout the duration of packet dumping and saved the outputs. Netstat gives information of which port number is being used by which application when the command is executed.

2.1.1.3 Extracting necessary information from packet dump

We do not take all packets into consideration. We are only interested in HTTP packets (and only requests, not responses). So we have filtered out all other packets from the packet dump we generated at the first step. We took only three fields from each packet: time, originating port and full request URI. This gives a time-sequenced log of port numbers and URIs that a port tried to connect to.

2.1.1.4 Aggregating packet dump and netstat logs

We have so far obtained two separate mappings: *application vs. port number* from netstat logs, and *port number vs. URL* from packet dump. We aggregate these two maps to create a time-sequenced log of applications and the URLs each application tried to contact (The App-URL table).

2.1.2 Matching the URLs with Domain-blacklists

We search the URLs in the App-URL table for known malicious domains. If an application tries to connect to a rogue domain (URL), we flag it as a malware. We can also enrich our blacklist by adding other domains contacted by a flagged application.

These steps are discussed in detail in the following section.

2.2 Details of Malware Detection steps

Our first step is to create an App-URL table. In this table, each row of the table indicates an attempt to make an HTTP connection by any application. We store the time, the application's unique identifier (package name), and the URL which was contacted.

2.2.1 Creating the App-URL table

2.2.1.1 Packet dumping

We need to use a software for recording all incoming or outgoing traffic (packets) of the android device. This can be done using *Wireshark* [?] in a computer which is connected to the same local network of the android device.

Alternatively, we can use a similar application in the mobile device. We have used *Shark for Root* [?] for this purpose. A rooted device is not required for this step. Non-rooted devices can use other applications, such as *tPacketCapture*, which captures packets by creating a VPN and directing all traffic through the VPN. We captured packets for a specific amount of time. This step produces a packet dump (*.pcap*) file.

2.2.1.2 Netstat Logging

The packet dump does not directly detect which packet is originated from/destined for which mobile application. The system differentiates packets of different applications by port numbers (source port for outgoing packets or destination port for incoming packets). Hence, we need to know which ports were being used by which applications when the packet was captured. We used the UNIX tool *netstat* [**?**] to get the mapping between applications and port numbers at a specific time.

Since the packets are recorded for some duration of time and netstat gives the *port number vs. application* mapping for an instance of time (just when the command is executed), a single netstat output will not suffice. Therefore, we executed netstat periodically, while the packets were being recorded.

We used *ADB* [**?**] to communicate with the android device. To access the interactive shell of the device, *adb shell* was used. In our experiment, we connected the android device with a UNIX computer. Then we executed the shell script shown in Fig. 2.1 in the computer.

```
for i in {1..100}
do
     adb shell "
     su -c 'busybox netstat -pnt | grep tcp'
     " > netstat
     adb shell "date +%s" > netdump$i
     awk '{print $4 ":" $7}' netstat > netstattemp
     awk -F":" '{print $5 " " $6}' netstattemp>>netdump$i
     echo finished: $i
     sleep 1
done
```

Figure 2.1: Shell script used for netstat logging.

This script calls netstat 100 times, with 1 second interval in between. It filters just the necessary information (port numbers and corresponding pid/package names) from each netstat output, and saves them in separate files, along with the timestamp when the dump was taken. So after executing this script, we had 100 files (namely `netdump1, netdump2, ... netdump100`). A single netdump file is shown in Fig. 2.2.

Figure 2.2: A single netdump file

This step requires a rooted android device. Because, being a stripped down variant of linux, Android does not come with the *netstat* executable by default. So we used *Busybox*, a tool that allows execution of all standard UNIX commands in android. Busybox cannot be installed without super user permissions.

2.2.1.3 Extracting necessary information from packet dump

Packet dump (.pcap) contains comprehensive meta information about all packets, along with their contents. However, we are only interested in HTTP packets and only three fields of each packet. Pcap filtering can be accomplished by many different ways among which we used Wireshark.

We opened the pcap file in Wireshark. Then the following display filter was applied on the dump:

```
http && ip.src == X.X.X.X
```

Here, `X.X.X.X` is the IP address of the device. This was used to filter out the http responses. For now, we are only interested in requests.
We kept only the following columns in Wireshark:

- Time (in Seconds since epoch format)
- Src Port
- Full Request URI

Then we exported the displayed packets summary in a plain text file. In our experiment, we named the file `filtered.txt` (shown in Fig. 2.3).

Timestamp	Port #	URL
1414082186.261850	57001	http://www.quora.com/api/do_action_POST
1414082186.531015	47612	http://www.quora.com/
1414082187.769571	47614	http://qsc.is.quoracdn.net/-28ce1f6c6095d6c5.css
1414082187.770059	47615	http://qsc.is.quoracdn.net/-aeeaea065aef57c7.js
1414082192.439645	47621	http://qph.is.quoracdn.net/main-thumb-t-4052-50-khhbtngfzevs...
1414082240.246866	45830	http://api.duolingo.com/api/1/version_info
1414082240.286386	54574	http://api.duolingo.com/api/1/store/get_inventory
1414082240.287393	55690	http://api.duolingo.com/api/1/store/get_inventory
1414082277.182687	47634	http://www.memrise.com/api/auth/facebook/
1414082279.105752	47635	http://www.memrise.com/api/app/settings/
1414082279.671243	47636	http://www.memrise.com/api/level/get/?with_content=true&lev...
1414082280.704813	47637	http://www.memrise.com/api/user/courses_learning/?user%5Fid...
1414082284.491800	47275	http://static.memrise.com/uploads/things/audio/14218347_136...
1414082284.491922	47276	http://static.memrise.com/uploads/things/audio/14218346_136...
1414082298.626474	54348	http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js
1414082302.333963	39488	http://data.flurry.com/aap.do
...		

Figure 2.3: Extracted information from the packet dump in `filtered.txt` file

2.2.1.4 Aggregating packet dump and netstat logs

Before this step, we had 100 files containing netstat outputs (*port-application* mapping at specific times). And we had a file filtered.txt, which contains the *port-URL* mapping for all HTTP request packets. We have written a script which processes all these files to produce the final App-URL table.

Since netdump files contains *port-app* mappings for specific moments (1 second apart), a packet's time will not necessarily match exactly with any of these moments. To assign such a packet to an application, we have made some assumptions.

Let t be the timestamp of a packet. Let $t_1,\ t_2,\ t_3, ..., t_{100}$ are the timestamps of the netstat outputs (they are stored in corresponding netdump files). Of course $t_1 < t_1 < t_3 < ... < t_{100}$. If $t < t_1$ or $t > t_{100}$, we discard the packet. We only consider packets with t such that $t_1 \leq t \leq t_{100}$.

Now for each of these packets, there is an i such that $t_i \leq t$ and $t_{i+1} > t$. We assign a packet to an application using the following rules:

1. If the same application A was using the packet's port at both t_i and t_{i+1}, then application A is the sender of the packet.

2. If application A was using the port at t_i, and the port was not in use at t_{i+1}, application A originated the packet.

3. If the port was not in use at t_i, and application A was holding it at t_{i+1}, application A originated the packet.

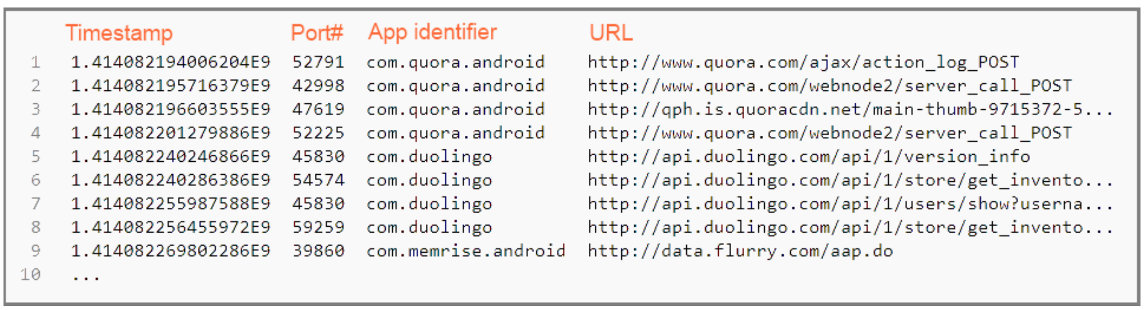
4. If the port was being used by application A at t_i and application B at t_{i+1} then,

   if $t - t_i \leq t_{i+1} - t$, application A originated the packet. Otherwise application B originated it.

5. If no application was using the port at either t_i or t_{i+1}, We discard the packet.

Case 5 indicates that after t_i, some application opened the port, sent some packet(s) and then released the port before t_{i+1}. So this packet has gone untraced. We can lessen the frequency of such occurrences by decreasing the interval between t_i and t_{i+1}.

So for every packet (except the ones of case 5), we know the app which originated it. And `filtered.txt` contains full request URI of all packets. So we now know the URL specified in the packet was contacted by this application. We have logged these (Application, URL) entries for each packet and the App-URL table is ready. A sample table is shown in Fig. 2.4.



	Timestamp	Port#	App identifier	URL
1	1.414082194006204E9	52791	com.quora.android	http://www.quora.com/ajax/action_log_POST
2	1.414082195716379E9	42998	com.quora.android	http://www.quora.com/webnode2/server_call_POST
3	1.414082196603555E9	47619	com.quora.android	http://qph.is.quoracdn.net/main-thumb-9715372-5...
4	1.414082201279886E9	52225	com.quora.android	http://www.quora.com/webnode2/server_call_POST
5	1.414082240246866E9	45830	com.duolingo	http://api.duolingo.com/api/1/version_info
6	1.414082240286386E9	54574	com.duolingo	http://api.duolingo.com/api/1/store/get_invento...
7	1.414082255987588E9	45830	com.duolingo	http://api.duolingo.com/api/1/users/show?userna...
8	1.414082256455972E9	59259	com.duolingo	http://api.duolingo.com/api/1/store/get_invento...
9	1.414082269802286E9	39860	com.memrise.android	http://data.flurry.com/aap.do
10	...			

Figure 2.4: Final output: App vs. URL table

2.2.2 Matching the URLs with Domain-blacklists

When the App-URL table is ready, the table can be sent to a central server. The server can search the table for already known malicious domains, and notify the android device of any rogue application which might be trying to connect to a black-listed domain. The server can also enhance its blacklist by adding new domains that are contacted by a malicious application.

2.3 Results

We analyzed two known malwares using this method: *DroidKungFu* and *Anserver-Bot*, both known for contacting remote C&C servers [?]. Within minutes of installation DroidKungFu accessed *www.waps.cn*, which was listed as a malicious domain in *virustotal.com*. Anserverbot did not contact any blacklisted domain within the first 10 minutes when we recorded packets. The reason might be using an unreliable and freely available domain-blacklist from the internet. Or worse, maybe it communicates over protocol(s) other than HTTP.

Chapter 3

System call based detection

In this chapter, we will discuss the overall strategy of our second approach for malware detection, which uses system call [explanation] traces of applications to predict malicious activity. Following chapters will discuss the implementation details of this model and also the results achieved in our experiment using this model.

The central idea is to run an application for a specific amount of time (in our simulation - 10 seconds). During its execution, the details about the system calls it make to the operating system are recorded. We developed a machine learning model/classifier that can detect malware based on its system call trace. The syscall records of known malwares and known non-malwares are used to train the classifier. There will be two phases: training and classification. How the known malware and non-malware traces will be used to train the classifier is described in section 3.1. Section 3.2 decribes how the model will classify an unknown app using its syscall trace.

3.1 Training

We will use a set of applications consisting both known malwares and known non-malwares as the training dataset. We will collect system call traces of all applications of the training dataset (all of the applications will be run for a specific amount of time). The system call trace of a single application is a list of system calls the application used during execution. For example: **{recv, semget, msgget, . . . }**; where **recv**, **semget**, **msgget** are system calls.

After collecting system call traces, We aggregate this traces to create two binary relation matrices M_{mal} and M_{nmal}. M_{mal} shows relation between system calls and malware applications, Where M_{nmal} shows relationship between system calls and non-malwares. M_{mal} and M_{nmal} matrices are defined as follows:

$$M_{mal}(i,j) = \begin{cases} 1 & \text{if } i^{th} \text{ malware uses } j^{th} \text{ syscall} \\ 0 & \text{otherwise} \end{cases}$$

$$M_{nmal}(i,j) = \begin{cases} 1 & \text{if } i^{th} \text{ non-malware uses } j^{th} \text{ syscall} \\ 0 & \text{otherwise} \end{cases}$$

Then we calculate the Goodness Rating of j^{th} syscall, G_j as follows,

$$G_j = \frac{1}{N_{nmal}} \sum_{i=1}^{N_{nmal}} M_{nmal}(i,j) - \frac{1}{N_{mal}} \sum_{i=1}^{N_{mal}} M_{mal}(i,j)$$

where N_{nmal} and N_{mal} are number of non-malware and malware samples.

3.2 Classification

To classify an unknown application as *malware* or *non-malware*, first, we execute the application for the same time duration duration we used with each training application. We collect the system call trace of that application during that execution, same as before. But this time, we also record the frequency of each syscall used by the application during execution. So now, the syscall trace of an application during classification phase can be expressed as a list of pairs of syscalls and their frequencies. For example: **{(recv,1032), (semget, 143), . . . }** is a trace of an application which called the **recv** routine 1032 times, **semget** 143 times and so on.

Then we define the Goodness Rating of that application as follows

$$G_{app} = \sum_{s \in S_{app}} G_s \times F_s$$

Where, S_{app} is the set of system calls used by *app* and F_s is the frequency of syscall s in *app*.

If we assume that malwares uses similar system calls which are distinctive from those used by non-malwares; It is logical to assume that a malware will use more syscalls those has lower goodness ratings and less syscalls having higher goodness ratings. The opposite can be said for non-malware applications. So this will result in higher goodness ratings of non-malware applications and lower goodness rating for malware applications.

Now for classification, we check if the goodness rating of the application under inspection exceeds some threshold. If so, we classify it as non-malware. otherwise we flag it as malware.

$$\begin{cases} \text{app is a malware} & \text{if } G_{app} > T \\ \text{app is not a malware} & \text{otherwise} \end{cases}$$

Where, T is a threshold. Theoretically, the threshold should be zero. But it actually depends on the experiment and the training data used. We will classify apps in validation dataset and calculate some metrics to evaluate our model. We will calculate the following metrics:

1. **Accuracy**

$$ACC = \frac{TP + TN}{P + N}$$

2. **Sensitivity/Recall** or **True Positive Rate**

$$TPR = \frac{TP}{P}$$

3. **Specificity** or **True Negative Rate**

$$SPC = \frac{TN}{N}$$

4. **Precision** or **Positive Predictive Value**

$$PPV = \frac{TP}{TP + FP}$$

5. **F-measure**

$$F = 2 \times \frac{precision \times recall}{precision + recall}$$

Chapter 4

Experiment on System call based classifier

In this chapter, we will go into details on the experiment we conducted to validate our model. The linklink.

4.1 Preparing Simulation

We collected system call traces of all applications using a single device. The reason behind this is we intended to provide identical environments for all applications to execute in. The device was reset to factory default configuration and the device needed to be *rooted* [explanation]. We used standard linux utility *strace*[explanation] to trace system call of applications. We also used *timeout* [explanation] command to run every application for a fixed duration of time. Although *strace* and *timeout* are standard linux utilities, they are not included in standard Android builds. So we had to collect the source code of this tools and cross-compile them for the architecture of the device on which the simulation is run. The compilation task requires *Android NDK* [explanation]. After the binaries are created for our desired CPU architecture (in our case **ARMv7**), they are put in the **/system/xbin** of our device, so that they can be accessed by a shell script run through *adb* [explanation]. Copying any binary into **/system** requires superuser permission, that is one of the reasons why we needed to *root* our device at the first place. We also need necessary drivers and android sdk installed on the host machine, where we will run the script.

4.2 Simulation

We planned to collect system call traces of a total of 453 malwares and 227 non-malwares. We wrote a batch script that automates the whole process. The script executes commands in the device using *adb* [explanation]. The workflow of the script is outlined in Algorithm 1.

Algorithm 1 Syscall trace collect script

```
procedure COLLECT-ALL-SYSCALL-TRACE(directory)
    for each apk file [explanation] in directory do
        pckgname ← get package name from that apk using aapt[explanation]
        Install the apk in the device.
        Launch the app
        pid ← ps(pckgname)
        stracelogs[pckgname] ← output of strace(pid) with 20 seconds timeout
        Force close the app
        Uninstall the app
    end for
    return stracelogs
end procedure
```

The exact script is given in linklink.

We have two directories, one containing 453 malwares apks and another containing 227 non-malware apks. The malware samples are collected from *Android Malware Genome Project* [explanation]. The non-malwares are directly downloaded from Google Play Store. We run the script twice. Once given the directory of malwares, and again for directory of non-malwares. After the execution, we are left with 453 malware trace files and 227 non-malware trace files. A sample single trace file is shown in figure 4.1.

```
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 29.08    1.285126        2824       455           semget
 26.44    1.168575         493      2371         6 recv
 17.94    0.793062      793062         1           wait4
  4.99    0.220610        1061       208           ioctl
  4.51    0.199257        3558        56           fsync
  4.23    0.186927         159      1176           msgget
  3.36    0.148469          81      1830           mprotect
  1.46    0.064444         198       325           write
  1.14    0.050596       10119         5           nanosleep
  1.14    0.050407         663        76         1 open
  0.85    0.037481         487        77           close
  0.67    0.029442         184       160           fstat64
  0.60    0.026524         144       184           mmap2
  0.50    0.021903         104       210           read
  0.47    0.020971         142       148           sigprocmask
```

Figure 4.1: System call trace of an application

4.3 Evaluating our model

We wrote a java program linklink which further processes these files and assess our model.

The program divides the trace files into two datasets, training and validation. 50 malwares and 50 non-malware traces are chosen randomly and put in the validation dataset. The rest of the traces are used to train the classifier. The details of the training and classification steps are described in the following sub-sections.

4.3.1 Training

The program aggregates all the traces in the training dataset and produce two relation matrices M_{mal} and M_{nmal}. M_{mal} and M_{nmal} are defined in the previous chapter. A sample relation matrix is shown in figure 4.2.

The two relation matrices are used to calculate the **Goodness rating**s of all syscalls.

	sigaltstack	mremap	rmdir	poll	pivot_root	pwrite	lstat64	bind	brk	pipe	getuid32	...
apps.ignisamerica.cleaner.pro.	1	0	1	1	0	0	1	0	0	1	0	
ar.com.moula.zoomcamerapro.	1	0	1	0	0	0	1	0	1	1	0	
ccc71.at.	1	1	0	0	1	0	0	0	1	1	0	
com.a0soft.gphone.acc.pro.	1	0	1	0	1	0	1	0	0	1	1	
com.adobe.reader.	0	0	1	1	1	0	1	1	1	0	0	
com.agilebits.onepassword.	1	0	0	0	0	1	1	1	1	0	0	
com.alarmclock.xtreme.free.	1	1	1	1	1	0	1	0	0	1	1	
com.anydo.	1	0	0	1	0	0	0	0	1	1	0	
com.appspot.swisscodemonkeys.bald.	1	0	1	1	1	1	1	0	0	0	0	
com.apusapps.browser.	0	0	1	0	1	0	1	1	0	1	0	
com.bdjobs.app.	1	1	1	0	1	0	1	0	0	0	0	
com.bikroy.	1	0	0	1	1	0	1	0	1	1	1	
...												

Figure 4.2: A sample relation matrix between syscalls and apps

4.3.2 Classification

Bibliography

[1] C. Perkins, D. Johnson, and J. Arkko, "Mobility support in IPv6," IETF RFC 6275, Jul 2011.

[2] V. Devarapalli, R. Wakikawa, A. Petrescu, and P. Thubert, "NEtwork MObility (NEMO) basic support protocol," RFC 3963, Jan 2005.

[3] S. Fu and M. Atiquzzaman, "SIGMA: A Transport Layer Handover Protocol for Mobile Terrestrial and Space Networks," *e-Business and Telecommunication Networks, Springer*, pp. 41–52, 2006.

[4] P. Chowdhury, M. Atiquzzaman, and W. Ivancic, "SINEMO: An IP-diversity based approach for network mobility in space," in *Second IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT)*, Pasadena, CA, Jul 17-21, 2006.

[5] H. Soliman, C. Castelluccia, K. El Malki, and L. Bellier, "Hierarchical Mobile IPv6 mobility management (HMIPv6)," IETF RFC 5380, Oct 2008.

[6] R. Stewart, Q. Xie, and M. Tuexen et al., "Stream Control Transmission Protocol (SCTP) dynamic address reconfiguration," IETF RFC 5061, Sep 2007.

[7] S. Fu and M. Atiquzzaman, "Handover latency comparison of SIGMA, FMIPv6, HMIPv6, and FHMIPv6," in *IEEE GLOBECOM*, St. Louis, MO, Nov 28-Dec 02, 2005.

[8] S. Fu and M. Atiquzzaman, "Hierarchical location management for transport layer mobility," in *IEEE GLOBECOM*, San Francisco, CA, Nov 27-Dec 1, 2006.

[9] A. S. Reaz, M. Atiquzzaman, and S. Fu., "Performance of DNS as location manager for wireless systems in IP networks," in *IEEE GLOBECOM*, St. Louis, MI, Nov 28 - Dec 2, 2005.

[10] S. Fu and M. Atiquzzaman, "Signaling cost and performance of SIGMA: A seamless handover scheme for data networks," *Wireless Communication and Mobile Computing*, vol. 5, no. 7, pp. 825–845, Nov 2005.

[11] A. S Reaz, P. K. Chowdhury, and M. Atiquzzaman, "Signaling cost analysis of SINEMO: Seamless End-to-End Network Mobility," in *First ACM/IEEE International Workshop on Mobility in the Evolving Internet Architecture*, San Francisco, CA, Dec 1, 2006.

[12] Christian Makaya and Samuel Pierre, "An analytical framework for performance evaluation of IPv6-based mobility management protocols," *IEEE Transactions on Wireless Communications*, vol. 7, no. 3, pp. 972–983, Mar 2008.

[13] Kumudu S. Munasinghe and Abbas Jamalipour, "Analysis of signaling cost for a roaming user in a heterogeneous mobile data network," in *IEEE Globecom*, New Orleans, LA, Nov 26-30, 2008.

[14] Jong-Hyouk Lee, Sri Gundavelli, and Tai-Myoung Chung, "A performance analysis on route optimization for Proxy Mobile IPv6," in *IEEE International Conference on Communications*, Dresden, Germany, Jun 14-18, 2009.

[15] Jiang Xie and Uday Narayanan, "Performance analysis of mobility support in IPv4/IPv6 mixed wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 2, Feb 2010.

[16] J. Xie and I.F. Akyildiz, "A novel distributed dynamic location management scheme for minimizing signaling costs in Mobile IP," *IEEE Transactions on Mobile Computing*, vol. 1, no. 3, pp. 163–175, Jul 2002.

[17] C. Perkins, D. Johnson, and J. Arkko, "Mobility support in IPv6," IETF RFC 6275, Jul 2011.

[18] R. Stewart, Q. Xie, and M. Tuexen et al., "Stream Control Transmission Protocol (SCTP) dynamic address reconfiguration," IETF RFC 5061, Sep 2007.

[19] C.A. Santivanez, B. McDonald, I. Stavrakakis, and R. Ramanathan, "On the scalability of Ad hoc routing protocols," in *IEEE INFOCOM*, New York, NY, June 23-27, 2002.

[20] Sumesh J. Philip, Joy Ghosh, Swapnil Khedekar, and Chunming Qiao, "Scalability analysis of location management protocols for Mobile Ad hoc Networks," in *IEEE WCNC*, Atlanta, GA, Mar 21-25, 2004.

[21] Lubna K. Alazzawi, Ali M. Elkateeb, Aiyappa Ramesh, and Waleed Aljuhar, "Scalability analysis for wireless sensor networks routing protocols," in *22nd International Conference on Advanced Information Networking and Applications*, Okinawa, Japan, Mar 25-28, 2008.

[22] Youngjune Gwon, James Kempf, and Alper Yegin, "Scalabilty and robustness analysis of Mobile IPv6, Fast Mobile IPv6, Hierarchical Mobile IPv6, and hybrid

IPv6 mobility protocols using a large-scale simulation," in *IEEE ICC*, Paris, France, Jun 20-24, 2004.

[23] Terho Hautala, Timo Braysy, Juha Makela, Janne Lehtomki, and Tommi Saarinen, "Scalability of mobility signaling in IEEE 802.11 WLAN," in *IEEE Vehicular Technology Conference*, Orlando, FL, Oct 6-9, 2003.

[24] Hoang Nam Nguyen and Iwao Sasase, "Downlink queuing model and packet scheduling for providing lossless handoff and QoS in 4G mobile network," *IEEE Transactions on Mobile Computing*, vol. 5, no. 5, pp. 452–462, May 2006.

[25] Mohsin Iftikhar, Tejeshwar Singh, Bjorn Landfeldt, and Mine Caglar, "Multi-class G/M/1 queueing system with self-similar input and non-preemptive priority," *Computer Communications*, vol. 31, pp. 1012–1027, Mar 2008.

[26] Mohsin Iftikhar, Bjorn Landfeldt, and Mine Caglar, "Towards the formation of comprehensive SLAs between heterogeneous wireless DiffServ domains," *Telecommunication Systems*, vol. 42, pp. 179–199, Dec 2009.

[27] Gunter Bolch, Stefan Greiner, Hermann de Meer, and Kishor S. Trivedi, *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*, Wiley-Interscience, New York, NY, 1998.

[28] "The network simulation - ns-2," http://www.isi.edu/nsnam/ns/.

[29] Harkirat Singh, Julan Hsu, Lochan Verma, Scott Seongwook Lee, and Chiu Ngo, "Green operation of multi-band wireless LAN in 60 GHz and 2.4/5 GHz," in *Consumer Communications and Networking Conference (CCNC)*, Las Vegas, NV, Jan 2011.

[30] Eldad Perahia, Carlos Cordeiro, Minyoung Park, and L. Lily Yang, "IEEE 802.11ad: defining the next generation multi-gbps Wi-Fi," in *7th IEEE Consumer Communications and Networking Conference (CCNC)*, Las Vegas, NV, Jan 2010.

[31] Ian F. Akyildiz, David M. Gutierrez-Estevez, and Elias Chavarria Reyes, "The evolution to 4G cellular systems: LTE-advanced," *Physical Communication*, vol. 3, pp. 217–244, Mar 2010.

[32] Sumit Singh, Raghuraman Mudumbai, and Upamanyu Madhow, "Distributed coordination with deaf neighbors: Efficient medium access for 60 GHz mesh networks," in *IEEE INFOCOM*, San Diego, CA, Mar 2010.

[33] Yi bing Lin, Wei ru Lai, and Rong jaye Chen, "Performance analysis for dual band PCS networks," *IEEE Transactions on Computers*, vol. 49, pp. 148–159, Feb 2000.

[34] Klaus Doppler, Carl Wijting, Tero Henttonen, and Kimmo Valkealahti, "Multi-band scheduler for future communication systems," *I. J. Communications, Network and System Sciences*, vol. 1, no. 1, pp. 1–9, Feb 2008.

[35] Lochan Verma and Scott Seongwook Lee, "Multi-band Wi-Fi systems: A new direction in personal and community connectivity," in *IEEE International Conference on Consumer Electronics*, Las Vegas, NV, Jan 2011.

[36] Donald Gross, John Shortle, James Thompson, and Carl M. Harris, *Fundamentals of Queueing Theory*, Wiley-Interscience, Aug 2008.

[37] Konstantin E. Avrachenkov, Nikita O. Vilchevsky, and Georgy L. Shevlyakov, "Priority queueing with finite buffer size and randomized push-out mechanism," *Performance Evaluation*, vol. 61, pp. 1–16, Jun 2005.

[38] Guido Appenzeller, Isaac Keslassy, and Nick McKeown, "Sizing router buffers," *Computer Communication Review*, vol. 34, pp. 281–292, Oct 2004.

[39] Gahng seop Ahn, Andrew T. Campbell, Andras Veres, and Li hsiang Sun, "Supporting service differentiation for real-time and best-effort traffic in Stateless Wireless Ad Hoc Networks (SWAN)," *IEEE Transactions on Mobile Computing*, vol. 1, pp. 192–207, Sep 2002.

[40] Kuo-Tay Chen, Szu-Lin Su, and Rong-Feng Chang, "Design and analysis of dynamic mobility tracking in wireless personal communication networks," *IEEE Transactions on Vehicular Technology*, vol. 51, no. 3, May 2002.

[41] Md. Shohrab Hossain, M. Atiquzzaman, and William Ivancic, "Survivability and scalability of space networks," in *NASA Earth Science Technology Forum*, Arlington, VA, Jun 22-24, 2010.

[42] Yin Fu Huang and Min Hsiu Chuang, "Fault tolerance for home agents in Mobile IP," *Computer Networks*, vol. 50, no. 18, pp. 3686–3700, Dec 2006.

[43] Jenn-Wei Lin and Joseph Arul, "An efficient fault-tolerant approach for Mobile IP in wireless systems," *IEEE Transactions on Mobile Computing*, vol. 2, no. 3, pp. 207–220, Jul-Sep 2003.

[44] J. Jue and D. Ghosal, "Design and analysis of replicated server architecture for supporting IP-host mobility," *ACM Mobile Computing and Communications Review*, vol. 2, no. 3, pp. 16–23, Jul 1998.

[45] J. Faizan Hesham EL-Rewini and M. Khalil, "Introducing reliability and load balancing in Mobile IPv6-based networks," *Wireless Communication and Mobile Computing*, vol. 8, no. 4, pp. 483–500, May 2008.

[46] Hui Deng, X. Huang, Kai Zhang, Zhisheng Niu, and Masahiro Ojima, "A hybrid load balance mechanism for distributed home agents in Mobile IPv6," *Personal, Indoor and Mobile Radio Communications*, pp. 2842–2846, Jan 2003.

[47] Romain Kuntz, Julien Montavont, and Thomas Noel, "Multiple mobile routers in nemo: How neighbor discovery can assist default router selection," in *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Poznan, Poland, Sep 15-18, 2008.

[48] Poul E. Heegaard and Kishor S. Trivedi, "Network survivability modeling," *Computer Networks*, vol. 53, no. 8, Jun 2009.

[49] S. Fu and M. Atiquzzaman, "Survivability evaluation of SIGMA and Mobile IP," *Wireless Personal Communications*, vol. 43, no. 3, pp. 933–944, Nov 2007.

[50] R. Wakikawa, V. Devarapalli, G. Tsirtsis, T. Ernst, and K. Nagami, "Multiple care-of addresses registration," IETF RFC 5648, Oct 2009.

[51] Jen-Yi Pan, Jing-Luen Lin, and Kai-Fung Pan, "Multiple care-of addresses registration and capacity-aware preference on multi-rate wireless links," in *International Conference on Advanced Information Networking and Applications*, Okinawa, Japan, Mar 25-28, 2008.

[52] Romain Kuntz, "Deploying reliable IPv6 temporary networks thanks to NEMO basic support and multiple care-of addresses registration," in *International Symposium on Applications and the Internet Workshops*, Hiroshima, Japan, Jan 15-19, 2007.

[53] Xiaohua Chen, Hongke Zhang, Yao-Chung Chang, and Han-Chieh Chao, "Experimentation and performance analysis of multi-interfaced mobile router scheme," *Simulation Modelling Practice and Theory*, vol. 18, no. 4, Apr 2010.

[54] Md. Sazzadur Rahman, Outman Bouidel, M. Atiquzzaman, and William Ivancic, "Performance Comparison between NEMO BSP and SINEMO," in *IEEE GLOBECOM*, New Orleans, LA, Nov 30-Dec 4 2008.

[55] Henrik Petander, Eranga Perera, Kun-Chan Lan, and Aruna Seneviratne, "Measuring and improving the performance of network mobility management in IPv6 networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 9, pp. 1671–1681, Sep 2006.

[56] "NEPL (NEMO Platform for Linux) howto," http://www.nautilus6.org/doc/nepl-howto/.

[57] R. Kuntz, "NEMO Basic Support implementation tests at the 6th IPv6 TAHI interoperability test event," http://www.nautilus6.org/doc/tc-nemo-tahi-interop-20050207-KuntzR.txt, Feb 2005.

[58] James Kempf, Jari Arkko, and Pekka Nikander, "Mobile IPv6 security," *Wireless Personal Communications*, vol. 29, pp. 398–414, Jun 2004.

[59] Dong Hu, Dong Zhou, and Ping Li, "PKI and secret key based mobile IP security," in *International Conference on Communications, Circuits and Systems*, Guilin, China, June 2006.

[60] Khaled Elgoarany and Mohamed Eltoweissy, "Security in Mobile IPv6: A survey," *Information Security Technical Report*, vol. 12, no. 1, pp. 32–43, Jun 2007.

[61] Errata Exist, S. Kent, and K. Seo, "Security architecture for the internet protocol," IETF RFC 4301, Dec 2005.

[62] C. Kaufman, P. Hoffman, Y. Nir, and P. Eronen, "Internet Key Exchange Protocol Version 2 (IKEv2)," IETF RFC 5996, September 2010.

[63] S. Kent, "IP Authentication Header," IETF RFC 4302, Dec 2005.

[64] S. Kent, "IP Encapsulating Security Payload (ESP)," IETF RFC 4303, Dec 2005.

[65] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Transaction on Information Theory*, vol. 22, no. 6, pp. 644–654, Nov 1976.

[66] T. Aura, "Cryptographically Generated Addresses (CGA)," IETF RFC 3972, March 2005.

[67] Greg O'Shea and Michael Roe, "Child-proof authentication for MIPv6 (CAM)," *ACM Computer Communications Review*, vol. 31, no. 2, Apr 2001.

[68] J. Harri, F. Filali, and C. Bonnet, "Mobility models for vehicular ad hoc networks: a survey and taxonomy," *IEEE Communications Surveys and Tutorials*, vol. 11, no. 4, pp. 19–41, Dec 2009.

[69] Christian Bettstetter, Hannes Hartenstein, and Xavier Prez-Costa, "Stochastic properties of Random Waypoint mobility model," *Wireless Networks*, vol. 10, no. 5, pp. 555–567, Sep 2004.

[70] Kuo-Hsing Chiang and Nirmala Shenoy, "A 2-d random-walk mobility model for location-management studies in wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 53, no. 2, Mar 2004.

[71] Tariq Ali and Mohammad Saquib, "Performance evaluation of wlan/cellular media access for mobile voice users under random mobility models," *IEEE Transactions on Wireless Communications*, vol. 10, no. 10, pp. 3241–3255, Oct 2011.

[72] Enrica Zola and Francisco Barcelo-Arroyo, "Probability of handoff for users moving with the random waypoint mobility model," in *IEEE Conference on Local Computer Networks*, Bonn, Germany, 2011.

[73] Plamen I. Bratanov and Ernst Bonek, "Mobility model of vehicle-borne terminals in urban cellular systems," *IEEE Transactions on Vehicular Technology*, vol. 52, no. 4, pp. 947–952, Jul 2003.

[74] G. Hosein Mohimani Farid Ashtiani Adel Javanmard and Maziyar Hamdi, "Mobility modeling, spatial traffic distribution, and probability of connectivity for sparse and dense vehicular ad hoc networks," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 4, pp. 1998–2007, May 2009.

[75] Amir Reza Momen and Paeiz Azmi, "A stochastic vehicle mobility model with environmental condition adaptation capability," *Wireless Communications and Mobile Computing*, vol. 9, no. 8, pp. 1070–1080, Aug 2009.

[76] Andrea Clementi, Angelo Monti, and Riccardo Silvestri, "Modelling mobility: A discrete revolution," *Ad Hoc Networks*, vol. 9, no. 6, pp. 998–1014, Aug 2011.

[77] Kahina Ait Ali, Mustapha Lalam, Laurent Moalic, and Oumaya Baala, "V-mbmm: Vehicular mask-based mobility model," in *9th International Conference on Networks*, Menuires, France, Apr 11-16, 2010.

[78] J. Chung and D. Go, "Stochastic vector mobility model for mobile and vehicular ad hoc network simulation," *IEEE Transactions on Mobile Computing (published online)*, Aug 2011.

[79] Francisco J. Martinez, Juan-Carlos Cano, Carlos T. Calafate, and Pietro Manzoni, "CityMob: A mobility model pattern generator for VANETs," in *IEEE ICC Workshops*, Beijing, China, May 2008.

[80] Amit Kumar Saha and David B. Johnson, "Modeling mobility for Vehicular Ad hoc Networks," in *ACM International Workshop on Vehicular Ad Hoc Networks (VANET)*, Philadelphia, PA, Oct 2004.

[81] Amit Jardosh, Elizabeth M. Belding-Royer, Kevin C. Almeroth, and Subhash Suri, "Towards realistic mobility models for mobile ad hoc networks," in *International Conference on Mobile Computing and Networking (MOBICOM)*, San Diego, CA, Sep 14-19, 2003.

[82] J. Harri, F. Filali, C. Bonnet, and Marco Fiore, "VanetMobiSim: Generating realistic mobility patterns for VANETs," in *ACM International Workshop on Vehicular Ad Hoc Networks (VANET)*, Los Angeles, CA, Sep 29, 2006.

[83] Jonghyun Kim, Vinay Sridhara, and S. Bohacek, "Realistic mobility simulation of urban mesh networks," *Ad Hoc Networks*, vol. 7, no. 2, pp. 411–430, Mar 2009.

[84] Hongyu Huang, Yanmin Zhu, Xu Li, Minglu Li, and Min-You Wu, "Meta: A mobility model of metropolitan taxis extracted from gps traces," in *IEEE Wireless Communications and Networking Conference*, Sydney, Australia, Apr 18-21, 2010.

[85] Md. Shohrab Hossain and M. Atiquzzaman, "Stochastic properties and application of city section mobility model," in *IEEE Global Communications Conference (GLOBECOM)*, Honolulu, HI, Nov 30-Dec 4, 2009.

[86] "Google Maps," http://maps.google.com.

[87] "City of New York: Department of Transportation," http://www.nyc.gov/html/dot/html/faqs/faqs_signals.shtml.

[88] S. Gundavelli and K. Leung et al., "Proxy Mobile IPv6," IETF RFC 5213, Aug 2008.

Appendix A

Acronyms

AH Authentication Header

AR Access Router

AZS Anchor Zone Server

BA Binding Acknowledgement

BSP Basic Support Protocol

BU Binding Update

CGA Cryptographically Generated Address

CN Correspondent Node

CoA Care of Address

CoT Care-of Test

CoTI Care-of Test Init

DDoS Distributed Denial of Service

DHCP Dynamic Host Configuration Protocol

DNS Domain Name System

DoS Denial of Service

ESP Encapsulating Security Payload

FTP File Transfer Protocol

HA Home Agent

HiSIGMA Hierarchical SIGMA

HIP Host Identification Protocol

HMIPv6 Hierarchical Mobile IP vesrion 6

HoA Home Address

HoT Home Test

HoTI Home Test Init

HZS Home Zone Server

ICMP Internet Control Message Protocol

IETF Internet Engineering Task Force

IKE Internet Key Exchange

IP Internet Protocol

IPsec IP security

LCoA Local Care of Address

LFN Local Fixed Node

LLM Local Location Manager

LM Location Manager

LMA Local Mobility Anchor

MAP Mobility Anchor Point

MH Mobile Host

MIP Mobile IP

MIPv6 Mobile IP vesrion 6

MITM Man In The Middle

MNN Mobile Network Node

MNP Mobile Network Prefix

MR Mobile Router

MSF Mobility Scalability Factor

NEMO NEtwork MObility

NRT Non-Real Time

PDA Personal Digital Assistant

RA Router Advertisement

RBU Refreshing Binding Update

RCoA Regional Care of Address

RR Return Routability

RT Real Time

RTT Round Trip Time

SA Security Association

SCTP Stream Control Transport Protocol

SIGMA Seamless IP-diversity based Generalized Mobility Architecture

SINEMO Seamless IP-diversity based Network Mobility

SPI Security Parameters Index

TCP Transmission Control Protocol

TNRL Telecommunications and Networks Research Lab

UDP User Datagram Protocol

VMN Visiting Mobile Node