In this paper, an automated technique is proposed to perform forensic analysis for new generation of mobile phones. Real-time information that resides on RAM is normally missed by forensic tools. The proposed technique is designed in a manner that benefits such volatile information to increase the rate of evidence acquisition. Same experiment is conducted manually by Android SDK tools to compare the results. Several scenarios of application monitoring were simulated. Our results showed 92.8% of successful acquisition when compared to the results of existing Android tools. The technique is then applied in a HTC Wildfire phone to capture and store live data into the memory card for further analyses. The results of the last experiment is then compared with a previous work and shows a better message capturing for short time interval data acquisition.

KEYWORDS
Forensics, Mobile phones, Android, Random Access Memory

1 INTRODUCTION
Most of standard techniques in forensic investigations involve gathering information from static storage in which data was recorded permanently. Even in case of deletion of data, forensic tools can still recover data from storage media [1]. In a closer look to information transactions, we see a vast amount of data being exchanged without being saved on permanent media such as messages in chat, or temporary web information. Current researches show that there are huge amounts of valuable information in live data transmission that are worth gathering for forensic investigations [2], [3]. Therefore, new methods are needed to put such data into consideration and help to improve the efficiency of standard forensic techniques [4].

Today we live in the era of smart mobile phones. Their capabilities are increasing and more integrated. With higher processing power and higher memory size, smart phones are becoming very powerful computing devices with almost the same features of normal PCs. By this development people’s trend to replace their computing devices by smart phones are increasing. Now they can easily keep documents and edit them, connect to the Internet and check emails, update social network profiles, etc. This means that more confidential data are now available in a portable device with less defense security and more intrusion capability in case of multiple connectivity options.

All the mentioned features drive recent researches to focus on mobile phone forensics. Current standard methods of mobile phone forensics are mainly performed on static storage media including removable memory cards, integrated flash memories and even the SIM.
A few works moved toward dynamic data acquisition recently that are introduced in the following sections. This paper is also going to propose a system that helps gathering live data from mobile phones’ random access memory.

### 2 RELATED WORK

A data gathering tool that could acquire files residing on flash memory of the mobile phone has been introduced in [6]. The connection could be done only through a serial interface, RS-232C.

Another on-phone forensic tool is presented in [9] that could be implemented on the phone itself. The tool was able to gather information from a Symbian enabled device with version 7, and store the acquired data in a memory card. However, if some files were in use at the same time they could not be copied by the forensic tool. Similar work is done in [10] for Symbian mobile phones. The proposed tool was implemented on removable memory card. As the authors also presented, their experiment takes longer time when compared to previous works.

Williamson et al. [11] did a very interesting work in which they compared different forensic tools like MOBILEdit! Forensic or Oxygen Phone Manager and then presented their results by claiming some tools that have been proposed as efficient before are not able to acquire all previously mentioned data.

For Android OS which is the main focus of our paper, there are also some interesting studies. In [8] several existing tools for Android phones are presented such as Nandroid backup and Android Debug Bridge, and their capabilities are discussed. An analysis of live transactions is done in [12]. The analysis is performed on Yahoo and Gtalk Messengers for iPhone. They tested the possibility of message recovery. Their experiment was valuable because the investigation was done for live data and could find some evidence for Yahoo messenger but not for Gtalk.

### 3 STATIC CONVERSION OF DYNAMIC INFORMATION

In this section, our technique which is able to acquire live data and map them to static media is presented. For the first two scenarios an emulator is used to test the system. For the third scenario a HTC wildfire device is used for real-time data acquisition through device’s random access memory (RAM).

The total system feature and components are shown in Table 1.

<table>
<thead>
<tr>
<th>Tools and Components</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System</td>
<td>Android 2.3</td>
</tr>
<tr>
<td>SDK Tools</td>
<td>Android Debug Bridge</td>
</tr>
<tr>
<td></td>
<td>Android Monkey</td>
</tr>
<tr>
<td></td>
<td>DDMS</td>
</tr>
<tr>
<td></td>
<td>hprof-conv</td>
</tr>
<tr>
<td></td>
<td>MAT</td>
</tr>
<tr>
<td>IDE</td>
<td>Eclipse</td>
</tr>
</tbody>
</table>

#### 3.1 Scenario 1: Manual memory acquisition using SDK tools

For this experiment we used Dalvik Debug Monitor Server (DDMS) tool from Android SDK package. By starting this tool a connection between DDMS and Android Debug Bridge (ADB) is established. On the other hand, a connected device provides a Virtual Machine (VM) monitoring service between ADB and DDMS. DDMS can monitor the processes running on the VM and is able to open a direct connection to the VM’s debugger by using the “adb” daemon. By assigning a port to the VM, the traffic can be forwarded to the debugger from the VM.
In this scenario we concentrated on Google mail application (GoogleMail-app) and performed a memory dump by the option provided in DDMS. Since the result of this dump is not a J2SE hprof standard format, we need to convert it to a form that can be used in a memory analyzer tool.

To do the conversion another Android tool should be used which is called “hprof-conv”. This conversion is necessary to make the output readable by the memory analyzers. The command is shown below.

```
~\AndroidSDK\tools>hprof-conv dump.hprof standard-dump.hprof
```

To check if the output is correct or not we tested it by the Memory Analyzer (MAT) in Eclipse. In both scenarios used in this study Android Monkey tool is used to generate random events for the application in order to fill up the memory space.

3.2 Scenario 2: Automated code for memory dumping using Eclipse IDE

To perform this experiment the Python language is selected to write a simple program in which we can call ADB commands for memory dumping. Since the language is not directly supported in Eclipse, a plug-in is used to make the development environment ready [13].

Hopefully all needed commands like getting root access, changing permissions and even killing processes was supported and performed well by ADB through the code. The result for the same Google application is generated at the end.

The result of this experiment shows 92.8% similarity when compared to the last scenario output. We considered this as an acceptable level of accuracy of our proposed technique and then it is used in the final scenario for live message acquisition.

3.3 Scenario 3: Live Message Acquisition through the proposed automated system

Now that we already tested the output of the proposed system with memory analyzer, another experiment is designed to test the capability of data acquisition by generating live chat messages to be captured from the random access memory. For this work a live chatting application [14] is used to generate messages between a mobile device and a PC. In total 20 messages are generated in different time intervals to check the performance and accuracy of the system. Results of this experiment are presented in Section 4.

4 RESULTS

The result of our last experiment shows that in short time intervals the acquisition rate is more accurate and reach to almost 100% of successful results. But when the time intervals become wider the acquisition rate decreases. On the other hand when using shorter time intervals the CPU usage increases that leads to the decrease in the performance of the mobile device. Therefore, there is no absolute conclusion on the practical feasibility of the system. However, what is obvious is that this technique can reduce the manpower and time for such forensic investigations.

Tables 2 and 3 show the results of different time intervals. As mentioned before the proposed system can capture almost all messages when the memory dumping process executes repeatedly and faster.

Table 2: Captured messages in long time intervals

<table>
<thead>
<tr>
<th>Time interval</th>
<th>Captured messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>30s</td>
<td>90%</td>
</tr>
<tr>
<td>60s</td>
<td>80%</td>
</tr>
<tr>
<td>90s</td>
<td>80%</td>
</tr>
<tr>
<td>120s</td>
<td>70%</td>
</tr>
</tbody>
</table>
Table 3: Captured messages in short time intervals

<table>
<thead>
<tr>
<th>Time interval</th>
<th>Captured messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>5s</td>
<td>100%</td>
</tr>
<tr>
<td>10s</td>
<td>100%</td>
</tr>
<tr>
<td>15s</td>
<td>95%</td>
</tr>
<tr>
<td>20s</td>
<td>95%</td>
</tr>
</tbody>
</table>

A previous work has been done in [15] to capture messages through real time traffic, including live messages. The capturing is done for both incoming and outgoing messages but we focus only on the incoming results. Although their system is quite more completed and even traffic generators with messages are written by the authors themselves but still it is the closest work we can compare our results since the authors performed message capturing in a part of their experiment. We compare our results with their work and it is shown in figure 1.

![Figure 1: The comparison for captured messages](image)

As the comparison shows the technique proposed in this paper can capture more messages with shorter time intervals. However, it is not as effective as previous work for long time intervals.

5 CONCLUSION AND FUTURE WORK

In this paper an automated technique is proposed to capture temporary data residing on random access memory. The sample output of the system is tested by standard tools provided in Android SDK package for its correction. The result showed the usability of the code. Then the program is used to capture live data, which are chat messages in our experiments, to show its effectiveness for being used as a forensic investigation tool.

Although the proposed system decreases the performance of the mobile device in terms of CPU usage, it could reduce human supervision in such investigations. As the results showed almost all presentable information in RAM can be dumped through the system if we do not consider its effects on the performance of the device.

Finally, our results were compared to previous work in [15], and showed a better message capturing in short time intervals.

There are several issues in the area of dynamic information forensics that can be considered as future work. One is to perform data acquisition in a way that has less impact on mobile device performance. Another future trend is to produce event acquisition systems that can be run on different Android operating systems without the need for reconfigurations.

REFERENCES