Applications’ Credentials Harvesting from Android Memory

Pasquale Stirparo

Sir, we may have a leak!

www.jrc.ec.europa.eu
Serving society
Stimulating innovation
Supporting legislation
$ whoami

✧ **Pasquale Stirparo**
  - Digital Forensics and Mobile Security Researcher at Joint Research Centre (JRC) – European Commission
  - Ph.D. candidate at KTH – Royal Institute of Technology, Stockholm

✧ Research Interests
  - Security and Privacy issues of **mobile communication protocols**
  - **Mobile Malware/Botnets**
  - **Digital Forensics**

✧ Background and previous experiences
  - Certified GCFA, OPST, OWSE, ECCE
  - Previously working as Pen-Tester and Digital Forensics Engineer
Android Memory Management

✧ Every app runs within a separate process, which has its own instance of the Dalvik-VM;

✧ When app is launched, the “Zygote” process is forked and Dalvik heap is preloaded with classes and data by Zygote

✧ Dalvik-VM implements Garbage Collection (GC) on the heap
Apps Analysed

Two Apps categories analysed

✧ Top downloaded apps
  • 11 of the most popular Android applications

✧ Mobile Banking apps
  • 15 mobile banking applications
Memory Acquisition...

✧ Acquisition with Linux Memory Extractor, a.k.a. LiME
  • Loadable Kernel Module
  • It allows full memory captures of linux-based devices, and so Android
  • Needs to be cross-compiled against device kernel source

$ adb push lime.ko /sdcard/lime.ko
$ adb forward tcp:4444 tcp:4444
$ adb shell
$ su
# insmod /sdcard/lime.ko “path=tcp:4444 format=lime”

On the destination computer
$ nc localhost 4444 > ram-dump.lime
... and Analysis

Volatility Framework

✧ Identify process id (PID) of the target application
  • `linux_pslist`, which gathers active tasks by walking the task struct->task list

✧ Map the process in the memory to find the heap offset
  • `linux_proc_maps`, which gathers process maps for linux

✧ Dump the heap
  • `linux_dump_map`, which writes selected memory mappings to disk
$ python vol.py --profile=LinuxGolfish-2_6_29x86 -f ebay.lime linux_pplist
Offset  Name          Pid    Uid     Gid     DTB        Start Time
----------  -------------------  ---  ---  ------  -------------------
0xca969400  com.ebay.mobile  379  10067  10067  0x0aec8000 2013-03-29 09:22:08

$ python vol.py --profile=LinuxGolfish-2_6_29x86 -f ebay.lime linux_proc_maps
-p 379 | grep heap

<table>
<thead>
<tr>
<th>Pid</th>
<th>Start</th>
<th>End</th>
<th>Flags</th>
<th>Pgoff</th>
<th>Major</th>
<th>Minor</th>
<th>Inode</th>
<th>File</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>379</td>
<td>0x0000b000 0x003d1000</td>
<td>rw-</td>
<td>0x0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0 [heap]</td>
<td>/dev/ashmem/dalvik-heap</td>
<td></td>
</tr>
<tr>
<td>379</td>
<td>0x409b2000 0x42124000</td>
<td>rw-</td>
<td>0x0</td>
<td>0</td>
<td>7</td>
<td>353</td>
<td>/dev/ashmem/dalvik-heap</td>
<td></td>
<td></td>
</tr>
<tr>
<td>379</td>
<td>0x42124000 0x449b2000</td>
<td>---</td>
<td>0x1772000</td>
<td>0</td>
<td>7</td>
<td>353</td>
<td>/dev/ashmem/dalvik-heap</td>
<td></td>
<td></td>
</tr>
<tr>
<td>379</td>
<td>0x46e02000 0x46e03000</td>
<td>r--</td>
<td>0x0</td>
<td>0</td>
<td>7</td>
<td>368</td>
<td>/dev/ashmem/SurfaceFlinger read-only heap</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$ python vol.py --profile=LinuxGolfish-2_6_29x86 -f ebay.lime linux_dump_map
-s 0x0000b000 --dump-dir /memdump/ebay-heap/

<table>
<thead>
<tr>
<th>Task</th>
<th>VM Start</th>
<th>VM End</th>
<th>Length</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>379</td>
<td>0x0000b000 0x003d1000 0x3c6000</td>
<td>/memdump/ebay-heap/task.379.0xb000.vma</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$ python vol.py --profile=LinuxGolfish-2_6_29x86 -f ebay.lime linux_dump_map
-s 0x409b2000 --dump-dir /memdump/ebay-heap/

<table>
<thead>
<tr>
<th>Task</th>
<th>VM Start</th>
<th>VM End</th>
<th>Length</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>379</td>
<td>0x409b2000 0x42124000 0x1772000</td>
<td>/memdump/ebay-heap/task.379.0x409b2000.vma</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
How do the findings look like?

Sample with identification labels

E3:1F40h: 61 74 61 3B 20 6E 61 6D 65 3D 22 75 73 65 72 6E ata; name="usern...
E3:1F50h: 61 6D 65 22 0D 0A 0D 0A 0D 0A 0D 0A 0D 0A 0D 0A 0D 0A 0D aperino........
E3:1F60h: 61 70 65 72 69 6E 70 61 73 77 6F 72 name="password"....paperopoli...

Data structure without identification labels

DD:B680h: 28 39 9B 40 00 00 00 00 1A 00 00 00 00 00 00 00 00 00 00 00 00 00 (9>@........
DD:B690h: 70 00 61 00 6F 00 6C 00 69 00 6E 00 6F 00 70 00 p.a.o.l.i.n.o.p.
DD:B6A0h: 61 00 70 00 65 00 72 00 69 00 6E 00 6F 00 3A 00 a.p.e.r.i.n.o:.
DD:B6B0h: 70 00 61 00 70 00 65 00 72 00 6F 00 70 00 6F 00 p.a.p.e.r.o.p.o.
DD:B6C0h: 6C 00 69 00 23 00 00 00 B8 30 9B 40 l.i.#...0>@
Forensics Implications

✧ Also **mobile memory analysis can be a goldmine** of useful information such as applications credentials

✧ Most applications save **credentials preceded by their identification label**

✧ **Processes still running after exiting the applications for 25 out of 26** applications analysed

✧ **LiME** it’s a kernel module → **low memory footprint**

✧ **Technique not yet completely mature** from forensics perspective
  
  • Invasive, requires to root target phone, requires precise kernel source code of the device

✧ **Need for kernel-agnostic module**
References


Joint Research Centre (JRC)

Web:  www.jrc.ec.europa.eu

Contacts: pasquale.stirparo@jrc.ec.europa.eu